Escuela Técnica Superior de
Ingeniería Informática
máster universitario en ingeniería
informática,
especialidad en ciberseguridad

Trabajo de Fin de Máster

Entorno de análisis de memoria volátil para estudiantes

Volatile memory analysis environment for students

Universidad de Málaga
Septiembre de 2019

Realizado por: Javier Ordoñez Martín
Tutorizado por: Ana Nieto Jiménez y Fco. Javier López Muñoz
Departamento: Lenguajes y Ciencias de la Computación

Resumen

La informática forense y concretamente el análisis de memoria volátil, se encuentran en su punto álgido debido al enorme número de incidentes tecnológicos que rodean a las grandes empresas, cada vez más necesitadas de profesionales formados en este complejo campo. También ha contribuido el hecho de que el análisis de memoria puede ayudar a otras disciplinas como el análisis de malware.

El objetivo de este Trabajo Fin de Máster (TFM) es facilitar el aprendizaje de los nuevos profesionales que se adentran en la informática forense, concretamente en el análisis de memoria volátil. Con este fin, se aporta un enfoque distinto al de las herramientas actuales. Concretamente se ha realiza un análisis de las herramientas actuales, identificando la necesidad actual de contar con una solución enfocada en asentar las bases de conocimiento de usuarios no expertos, sirviéndole de apoyo en su periodo de formación.

La herramienta desarrollada, *Vol-E* basada en *Volatility*, cuenta con dos modos de uso, permitiendo introducir al usuario (el alumno) en casos de análsis de memoria sin necesidad de conocer complicados comandos o abrumarse por respuestas complejas gracias a su intuitiva interfaz gráfica. Consiste en enlazar el análisis de memoria volátil con un incidente tecnológico conocido, despertando el interés del usuario y facilitando la asimilación de conceptos gracias a la utilización de preguntas cortas sobre el incidente. Por otro lado, la herramienta también permite al alumno realizar un análisis guiado de memoria volátil, dividiendo el análisis en 3 grupos de conocimiento: análisis de procesos, análisis de red y evidencias extraíbles. De esta forma esta herramienta prepara a usuarios que comienzan su formación para que conozca de forma amena los elementos fundamentales en el análisis de memoria.

Palabras clave

Memoria volátil, procesos, red, malware, JSON, LaTeX, clave de registro análisis de memoria, informática forense, Volatility.

Abstract

Computer forensic and, more specifically, volatile memory analysis, is at a decisive point because of the high number of technological incidents affecting the big companies, which are in need of highly qualified professionals in this complex area. The fact that the memory analysis can be helpful to other disciplines like malware analysis is also contributing to it.

The main goal of the project is to make the learning process of students starting in computer forensic, and specifically in volatile memory, easier. For this purpose, a new approach is proposed, different from the ones the current tools provide us with. More specifically, an analysis of the current tools has been conducted, identifying the current need of having a solution focused on settle the knowledge of non-expert users, supporting their training period.

The developed tool, *Vol-E* based on *Volatility*, can be used in two ways, allowing the user (the student) to start with memory analysis scenarios without knowing complicated commands or getting frustrated because of complex answers thanks to its intuitive graphic user interface. It consists on linking volatile memory analysis with a well-known technological incident, increasing the user's interest on the matter and making the process of learning the commands easier by making short questions about the incident. Besides that, the tool also allows the student to conduct a guided analysis of the volatile memory by dividing the analysis in three knowledge groups: process analysis, network analysis and removable evidences. This way, the tool makes the users able to start their education in the area by teaching them the base concepts of the memory analysis in a pleasant way.

Keywords

Volatile memory, process, network, computer forensics, malware, register key, JSON, LaTeX, Volatility.

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

En un mundo donde ya existen más de 22 mil millones de dispositivos digitales conectados a *Internet*, donde la tecnología y el avance imparable de los dispositivos digitales y su implantación en la sociedad actual, tanto por particulares como por grandes empresas, y ante la creciente cantidad de amenazas e incidentes digitales, surge la necesidad de realizar informes y análisis forenses de dichos dispositivos. Lo cierto es que como profesionales de ciberseguridad debemos enfrentarnos al nuevo reto de afrontar la gran cantidad de incidentes digitales de esta década, y de preparar a los distintos expertos que serán necesarios para entenderlos.

Gran parte de la información sensible de lo ocurrido en un incidente tecnológico se encuentra en la memoria volátil. Una memoria efímera, que requiere de profesionales cualificados en el análisis y extracción de evidencias, siendo esta una ciencia que, dependiendo del escenario, puede ser muy compleja.

Debido a la gran demanda del mercado de profesionales especializados en ciberseguridad para la prevención y el análisis de incidentes de seguridad, surgen nuevas necesidades formativas. Uno de los problemas con las que se pueden topar los formadores y también los profesionales que se inicien en este mundo por cuenta propia es la dificultad de encontrar herramientas intuitivas que no supongan un alto coste económico y que les permitan formarse adecuadamente.

De hecho, como se comprobará durante este trabajo, existen en el mercado numerosas herramientas tanto gratuitas como de pago para gran cantidad de sistemas operativos, que permiten realizar un análisis detallado de la memoria volátil en busca de evidencias digitales. El principal problema reside en que dichas herramientas en su mayoría tienen un coste muy elevado por licencia y en el caso de las gratuitas no proveen de una interfaz gráfica intuitiva o requieren que el usuario final posea cualidades técnicas en el ámbito de la informática forense.

No es un tema baladí formar a la gran cantidad de forenses digitales que el mercado necesita. Esta además es una disciplina que ya se está estudiando en diversas

carreras no todas ellas técnicas, precisamente, porque se necesita la cooperación entre diversos tipos de expertos para ofrecer solución a los casos actuales. Para esta formación se hace necesaria la creación de nuevas herramientas pensada en ellos, en la gran cantidad de personas que se encuentran formándose, personas con pocos o escasos conocimientos en el análisis de memoria volátil. También es importante que estas herramientas proporcionen otra forma de visualizar el análisis, más cercano al lenguaje común, para que los expertos en ciberseguridad puedan conocer otras formas de llegar a otro tipo de expertos.

La herramienta desarrollada durante este Trabajo Fin de Máster (TFM) ha sido diseñada para reforzar los conocimientos, asentar las bases y permitir realizar un primer análisis de memoria volátil, dejando a un lado el conocimiento en herramientas complejas, y centrando al individuo en lo realmente importante, los hechos.

1.1. Objetivos y Metodología

Los objetivos de este trabajo fin de máster son los siguientes:

Objetivo 1 Analizar las necesidades actuales del mercado con respecto al análisis de memoria volátil, visto desde la perspectiva de un estudiante o persona que se está iniciando en el análisis forense. Determinar la viabilidad de realizar una mejora sobre los sistemas actuales a fin de proveer a los estudiantes de una herramienta que sirva de aprendizaje.

Objetivo 2 Realizar un análisis de las necesidades principales en el análisis de memoria volátil, conjunto de funcionalidades más utilizadas y zonas de la memoria volátil donde se pueden encontrar evidencias criticas a fin de realizar con esta información un conjunto de funcionalidades de fácil uso para usuarios no expertos.

Objetivo 3 Desarrollar una herramienta de análisis de memoria volátil para sistemas operativos Windows que dispondrá de una interfaz gráfica intuitiva que sirva de apoyo en el ámbito de la informática forense. Sirviendo no solo para la realización de análisis forenses si no como herramienta de aprendizaje para futuros profesionales. Permitiendo la integración con otras herramientas ya existentes en el mercado y que se encuentran bajo licencia GNU, como es el caso de *Volatility* [2] y *John the Ripper* [3].

Objetivo 4 Documentar las funcionalidades implementadas a fin de que puedan servir de apoyo en una futura expansión del trabajo realizado.

Con el fin de conseguir los objetivos anteriormente descritos, se ha utilizado una metodología ágil, incremental e iterativa.

Lo cual ha permitido:

- Mejorar la calidad, minimizando los errores en cada entregable.
- Mejorar la productividad, la organización en hitos y entregas permiten una mejor planificación lo que repercute directamente en la productividad.

Dividiéndose el trabajo en varios hitos o fases los cuales tienen asociados un conjunto de entregas:

- Hito 1: Análisis del mercado, identificación de carencias de los productos actuales.
- Hito 2: Realización de un estudio de las necesidades principales en el análisis de memoria volátil.
- Hito 3: Implementación de una herramienta de análisis de memoria volátil para sistemas operativos Windows. La cual deberá incluir el conjunto de funcionalidades extraídas del hito 2. Adicionalmente deberá disponer de una interfaz gráfica intuitiva que presentaran estas funcionalidades al usuario final.
- Hito 4: Implementación de una funcionalidad que permita servir como ayuda en la formación de futuros profesionales que deberá estar integrada en la herramienta del hito 3 . Esta herramienta estará enfocada en el ámbito académico y formativo, permitiendo formar, reforzar y validar los conocimientos adquiridos por los usuarios.

El conjunto de entregables asociados a estos hitos se encuentran dentro de los apartados de este documento.

1.2. Tecnologías y herramientas

El desarrollo y documentación de este trabajo fin de máster engloba una gran cantidad de tecnologías y herramientas distintas.

Debido a la complejidad del proyecto y a fin de poder seguir una metodología ágil e iterativa se ha utilizado la herramienta en linea *Trello*[4] el cual es un gestor de tareas basadas en el método *KANBAN*. Es una herramienta colaborativa que organiza proyectos en tablones, permitiendo de un solo vistazo ver que tareas están pendientes, cuales en proceso y cuales terminadas. Facilitando por tanto la gestión y mantenimiento de tareas del proyecto.

La herramienta final del trabajo fin de máster ha sido desarrollada en *Python*[5] versión 3.7 el cual es un lenguaje extendido actualmente dentro de la ingeniería del Software ya que es extremadamente flexible y permite ser portado a sistemas operativos tanto Windows, Linux como Mac. Otro de los motivos por los que se ha decidido usar *Python* se debe a la gran interconexión con la herramienta *Volatility*, también desarrollada en *Python* la cual nos proveerá de una gran cantidad de funcionalidades para el análisis de memoria volátil. *Volatility* sera la base sobre la que se implementaran las funcionalidades a medida que requerirá nuestra herramienta.

Se ha utilizado *Tkinter*[6] para el desarrollo de la interfaz gráfica de la herramienta ya que, es el estándar de facto de *Python* para las interfaces gráficas de usuarios.

Adicionalmente para el diseño de la interfaz gráfica se ha utilizado la herramienta en linea *Balsamiq Mockups*[7] la cual facilita y agiliza la creación de bocetos.

Ante la necesidad de realizar un desencriptado de las claves obtenidas en una de las funcionalidades implementadas, se ha utilizado la herramienta *John the Ripper* de criptografía que aplica fuerza bruta para descifrar contraseñas. La decisión de utilizar esta herramienta y no otra se basa en que *John the Ripper* es capaz de romper una gran cantidad de algoritmos en poco tiempo siendo además de código abierto.

La memoria del proyecto ha sido desarrollada en *LaTeX*[8] debido a su versatilidad en la creación de documentación.

1.3. Estructura de la memoria

La memoria de este trabajo fin de máster se ha dividido en los siguientes apartados:

- Introducción: En la introducción se realiza un resumen de la descripción del trabajo, las herramientas y tecnologías que han sido necesarias para la realización del trabajo fin de máster. También se describe la metodología y fases del trabajo.
- Análisis de soluciones existentes: En este apartado se realizara un análisis de las herramientas que existen actualmente en el mercado y las necesidades no cubiertas por estas. Permitiendo descubrir, desde el punto de vista de un estudiante o persona no avanzada en la materia del análisis forense, los puntos que las herramientas actuales no cubren.
- Diseño de la solución propuesta: Una vez conocidas las necesidades de nuestro publico objetivo, se ha realizado un análisis y diseño de las funcionalidades que deberemos incluir en nuestro proyecto a fin de abarcar las necesidades encontradas en el apartado Análisis de soluciones existentes.

- Desarrollo de la solución propuesta: En este apartado se muestra el desarrollo e implementación de la herramienta. Los distintos modos de uso y sus funcionalidades. Además, cuenta con la información necesaria sobre las dependencias del software desarrollado con respecto a otras herramientas y/o bibliotecas.
- Validación: Este apartado incluye las validación sobre el desarrollo realizado en el apartado anterior.
- Conclusiones y futuras implementaciones: En este apartado se han expuesto las conclusiones, reflexiones y problemas encontrados a la hora de realizar este trabajo de fin de máster, así como futuras ampliaciones que se pueden realizar tomando como base este trabajo.

Capítulo 2

Análisis de soluciones existentes

Este proyecto surge de la necesidad de dar otro tipo de enfoque al análisis de memoria volátil. Dirigiéndolo a usuarios con escaso conocimiento en la materia o que estén en fase de aprendizaje.

En la actualidad el análisis de memoria volátil está a la orden del día, ya que vivimos en un mundo altamente tecnológico donde la seguridad, los procesos judiciales y los sistemas informáticos se entremezclan. Es por ello que, existen una gran cantidad de herramientas que facilitan el análisis de memoria volátil, cada una con una perspectiva distinta.

Desde el inicio de la informática forense en el año 1980 hasta la actualidad, multitud de herramientas han copado el mercado, concretamente en el análisis de memoria volátil.Por lo que, resulta necesario realizar un análisis de las herramientas que actualmente hay en el mercado, permitiendo conocer el alcance, objeto, ventajas e inconvenientes de estas.

Una de las herramienta más utilizadas y populares es *Volatility*, publicada en 2007, desarrollada en *Python* permitiendo ser usada en sistemas operativos *Windows*, *Linux* y *Mac Os*, cuenta con una gran cantidad de extensiones. Permite analizar procesos, redes y buscar código malicioso.

Volatility permite consultar los servicios que se estaban ejecutando en el sistema y extraer información de ficheros para realizar un análisis posterior. Gracias a todo esto y debido a que cuenta con una gran cantidad de funcionalidades y una licencia GNU la convierte en una de las herramientas más utilizada del mercado.

Una de las principales desventajas que tiene la herramienta *Volatility* se debe a que no dispone de una interfaz gráfica que ayude y facilite su uso. En un intento de solucionar esto, apareció la herramienta *Volatility Workbench*[9] la cual provee de interfaz gráfica a la *api* de *Volatility*, permitiendo ejecutar la funcionalidad de esta sin necesidad de recordar los parámetros necesarios.

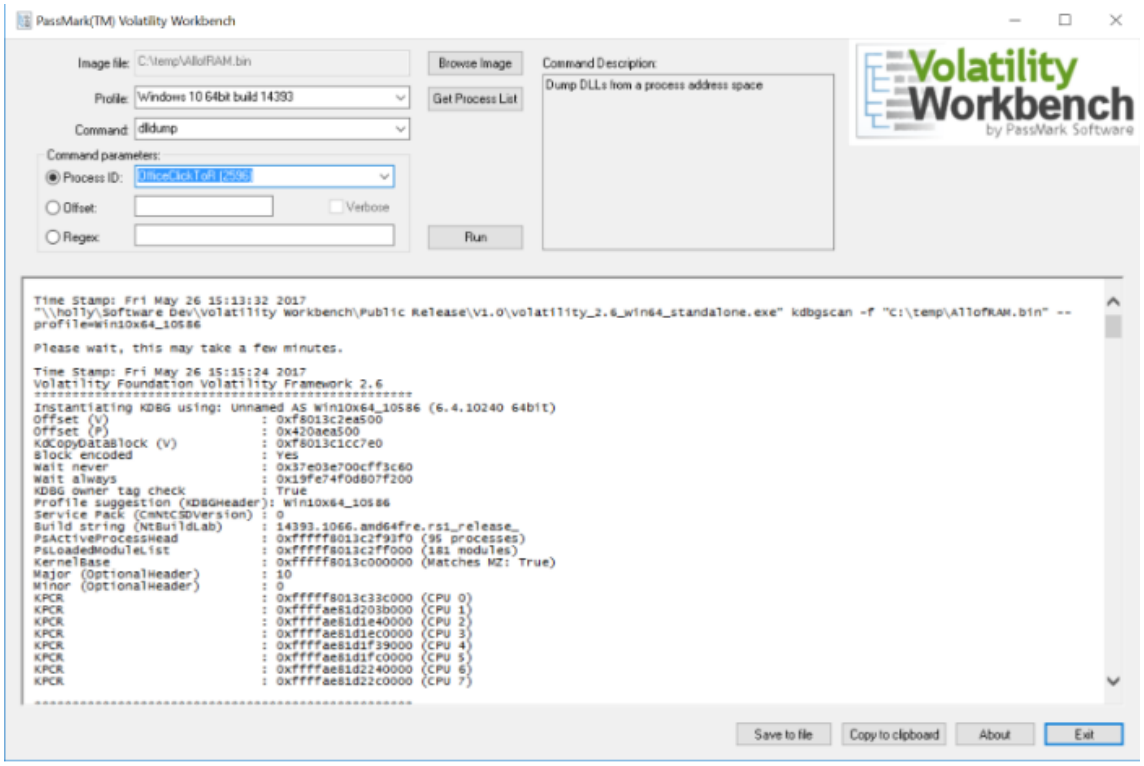

Figura 2.1: Interfaz gráfica *Volatility Workbench*.

Actualmente solo está disponible para sistemas operativos Windows y únicamente es capaz de realizar análisis de memoria Linux o Mac OS. Permite almacenar el perfil de la memoria que se está analizando, lo cual mejora el rendimiento en futuras consultas. Contiene un sellado de tiempo al ejecutar los comandos permitiendo establecer la cronología del análisis.

Otro de las principales herramientas utilizada únicamente en sistemas operativos Windows es *Windows Scope*[10].

Uno de sus usos más destacados es la detección e ingeniería inversa de *rootkits* y otros programas maliciosos. Adicionalmente permite la adquisición y el análisis de computadoras con Windows XP a través de Windows 10 representando el conjunto de procesos, DLLs y controladores que se estaban ejecutando en la computadora en el momento de la captura de la memoria, así como *sockets* de red abiertos, identificadores de archivos e identificadores de claves de registro. Proporciona gráficos de flujo de control y desmontaje para código ejecutable, además de una versión *live* para dispositivos móviles.

Frente a esta se encuentra *OsForensics*[11], la cual es es una suite de herramientas de pago que cuenta con una interfaz gráfica, permite el análisis de memoria volátil

utilizando como base *Volatility* para el análisis y extracción de evidencias.

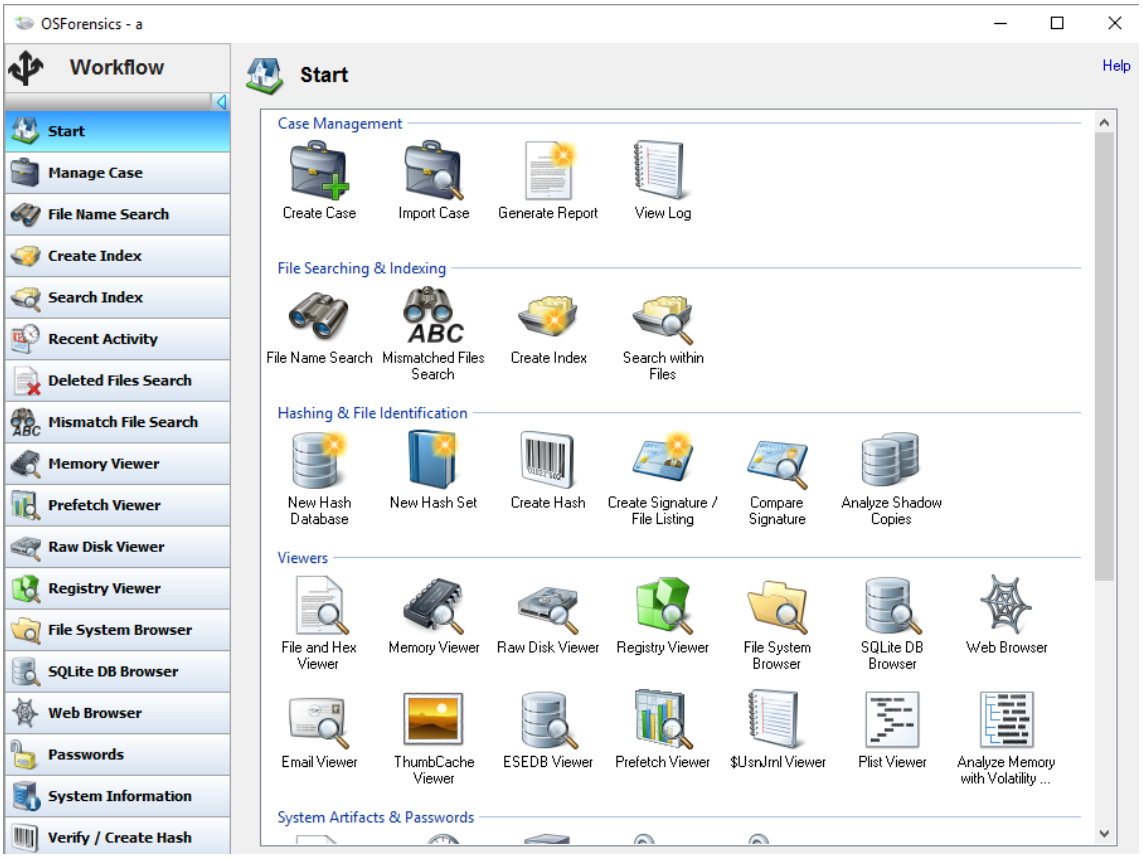

Figura 2.2: Interfaz gráfica *OsForensics*.

A diferencia de otras herramientas, *OsForensics*, permite realizar búsqueda de hash en base de datos, búsqueda por índices y búsqueda de archivos por nombre.

En el ámbito de las herramientas de software libre, se debe destacar *Memoryze*[12]. Esta herramienta es capaz de enumerar todos los procesos en ejecución, informando todos los identificadores abiertos de un proceso, enumerar el espacio de direcciones virtuales mostrando todas las DLLs cargadas, las partes asignadas del montón y la pila de ejecución. Permite enumerar todos los *sockets* de red que un proceso tiene abiertos, incluidos los que han sido ocultados por *rootkits*. Incluyendo soporte para sistemas operativo Windows y Mac.

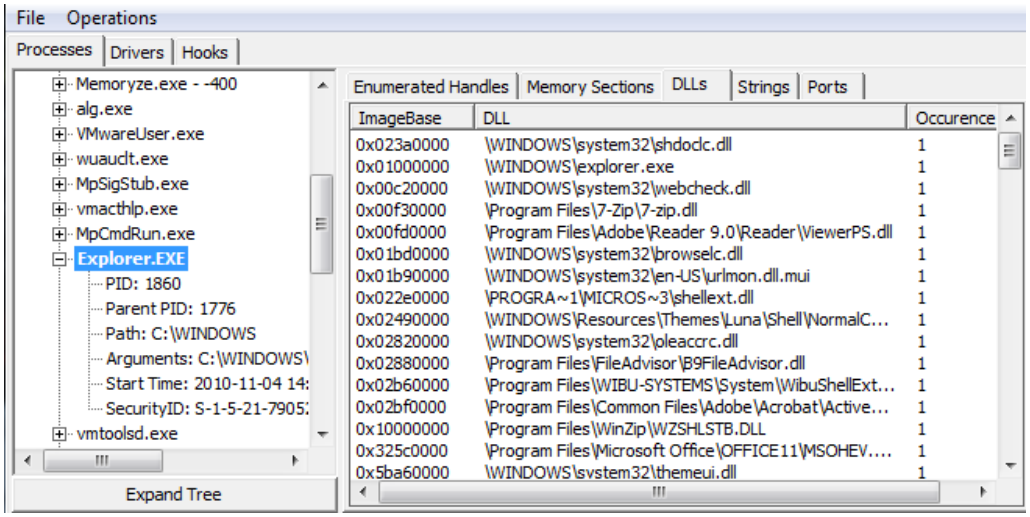

Figura 2.3: Interfaz gráfica *Memoryze*.

Perteneciente al conjunto de herramientas desarrolladas por la empresa Fireeye, al igual que *Memoryze* se encuentra *RedLine*[13].

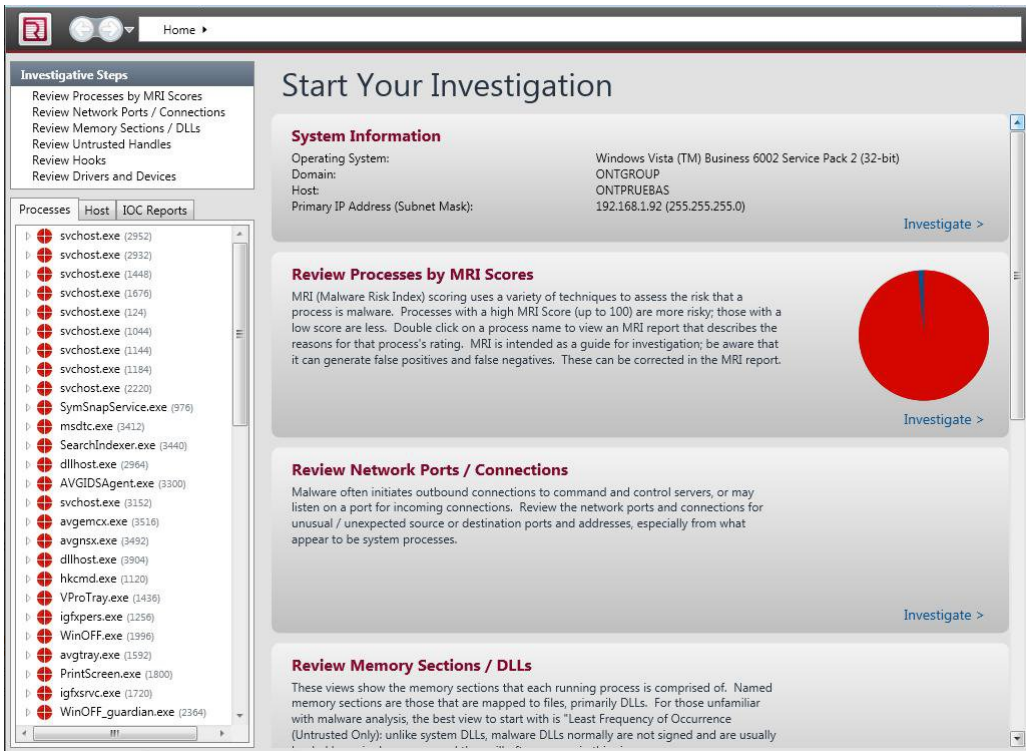

Figura 2.4: Interfaz gráfica *Redline*.

RedLine pertenece a un conjunto de herramientas conectadas entre sí, siendo capaz de realizar capturas de memoria volátil para posteriormente realizar un análisis a través de un entorno gráfico versátil y fácil de usar. Como elemento diferenciar con respecto a las otras herramientas *Redline* permite realizar un análisis de indicadores de compromiso (IOC). Tal y como se puede observar en la Figura 2.4

El uso indicadores de compromiso en el análisis permite un intercambio sencillo de información con otras entidades y herramientas para realizar una gestión eficiente y rápida del incidente.

Por otro lado, esta la herramienta de pago *Responder PRO*[14], la cual permite realizar un análisis de memoria volátil basado en el comportamiento.

Permitiendo realizar captura la memoria y análisis aprovechando el motor de comportamiento patentado por *Digital DNA*, para obtener una puntuación de impacto, que ayuda a los usuarios en el análisis de malware. Adicionalmente es capaz de escáner los fragmentos de documentos, el historial de Internet y realizar una extracción automática de claves y contraseñas almacenadas en la memoria.

Otro de los elementos diferenciadores de esta herramienta con respecto a otras es que permite el desmontaje automatizado de código, informes de perfil de comportamiento, búsqueda de patrones y representación de gráficos de flujo de control.

Las siguientes tablas comparativas (Tabla 2.1, Tabla 2.2 y Tabla 2.3) permite comprobar las diferencias principales entre las herramientas analizadas:

Tabla 2.1: Resumen de aplicaciones I

Herramientas/Características	**Volatility**	**Volatility Workbench**	**Windows Scope**
Licencia	Software libre	Software libre	Software de pago
SO Soportado	Windows, Linux, Mac OS	Windows 10 y 7	Windows 10
Interfaz Gráfica	No	Sí	Sí
Captura de memoria Volátil	No	No	Sí
Análisis de procesos	Sí	Sí	Sí
Análisis de conexiones de red	Sí	Sí	Sí
Desensamblado de código PRO	No	No	No
Análisis de malware	Sí	Sí	Desconocido

Tabla 2.2: Resumen de aplicaciones II

Herramientas/Características	**Memoryze**	**OsForensics**
Licencia	Software libre	Software de pago
SO Soportado	Windows y Mac OS	Windows XP y Windows server 2000 y superior
Interfaz Gráfica	Sí	Sí
Captura de memoria Volátil	No	Sí
Análisis de procesos	Sí	Sí
Análisis de conexiones de red	Sí	Sí
Desensamblado de código PRO	No	No
Análisis de malware	Desconocido	Desconocido

Tabla 2.3: Resumen de aplicaciones III

Herramientas/Características	**Redline**	**Responder PRO**
Licencia	Software libre	Software de pago
SO Soportado	Windows	Windows 7 y Windows server 2008 o superior
Interfaz Gráfica	Sí	Sí
Captura de memoria Volátil	Sí	Sí
Análisis de procesos	Sí	Sí
Análisis de conexiones de red	Sí	Sí
Desensamblado de código PRO	Desconocido	Sí
Análisis de malware	Sí	Sí

Tras el análisis de las diferentes herramientas que existen en el mercado se pueden extraer los siguientes puntos:

- Existen gran cantidad de herramientas que dan soporte al análisis memoria volátil.
- Gran parte de las herramientas toman como base a *Volatility* ya que permite cantidad de extensiones, y soporta los principales sistemas operativos y es gratuita.
- El enfoque de estas herramientas requiere que los usuarios tengan conocimientos avanzados en la informática forense y el análisis de memoria volátil.

El análisis de las soluciones existentes en el mercado, constata que existe gran cantidad de herramientas, pero todas están enfocadas a usuarios con alto conocimiento.

A diferencia de las herramientas que existen actualmente, en este trabajo fin de máster se ha diseñado una herramienta enfocada al análisis de memoria volátil con una interfaz gráfica intuitiva, que se enfoca en consolidar conocimientos y servir como base para el aprendizaje del usuario, en este caso el alumno. Cubriendo, por tanto, ese vacío que existe actualmente para las herramientas de formación en ciberseguridad.

Para ello, la herramienta desarrollada se aleja de la utilización de comandos, poco intuitivos y con los que los usuarios no expertos suelen sentirse abrumados. Y, sin embargo, asesora sobre los pilares básicos del análisis de memoria y usa una herramienta tan extendida como Volatility.

Además, el desarrollo final, permite ser ejecutado en el mayor numero de sistemas operativos posibles, permitiendo así abarcar el mayor numero de usuarios finales.

En contra posición, debido a la necesidad de acotar el proyecto y establecer un alcance realista. La herramienta únicamente permite realizar análisis de memoria volátil de sistemas operativos Windows. El motivo por el cual se ha decidido este sistema operativo y no otro, es que los sistemas operativos Windows copan la cuto de marcado más alta para ordenadores, superando ampliamente el 70 % [15].

Capítulo 3

Diseño de la solución propuesta

3.1. Aspectos Generales

El análisis de la sección 2, revela que existe una necesidad incipiente en formar a nuevos profesionales. Desprendiéndose la necesidad de que la herramienta cuente con dos modos de uso, claramente diferenciados pero unidos entre si.

Modo historia: El modo historia tiene un enfoque educativo. Su objetivo es afianzar los conceptos de los alumnos, abstrayendo la utilización de comandos o funcionalidades de análisis de memoria volátil.

A fin de cumplir este propósito, se le mostrara al alumno una "historia"por la que ira avanzando para descubrir los secretos que el análisis de memoria va exponiendo.

Crea tu propia análisis: Este modo tiene como objetivo permitir al alumno realizar el análisis de memoria volátil.

A diferencia de la gran mayoría de las herramientas del mercado, el alumno no necesitara conocer que funciones existen en el análisis de memoria volátil, ni los parámetros que estas necesitan. Se le proveerá al alumno de un conjunto de funcionalidades principales sobre las que podrá sacar sus conclusiones en el análisis de memoria.

Estos dos modos de uso están englobados dentro de una misma herramienta, formando parte de un todo. Teniendo que cumplir los siguiente requisitos no funcionales:

- RNF1 - Almacenamiento de datos en ficheros: La solución propuesta debe permitir almacenar los datos de las historias en ficheros.

- RNF2 - Adaptable a varios Sistemas Operativos: La solución propuesta debe permitir al alumno utilizar la herramienta en los sistemas operativos Windows, Linux y Mac. Permitiendo de esta forma universalizar su utilización.
- RNF3 - Usabilidad: La solución propuesta debe proporcionar mensajes de error que sean informativos y orientados al usuario final. Poseyendo interfaces bien formadas.

A nivel general, la solución propuesta debe cumplir los siguientes requisitos funcionales:

- RF1 - Iniciar modo historia: El alumno debe poder iniciar el modo historia.
- RF2 - Crear análisis: El alumno debe poder realizar un análisis de memoria volátil.

Estos dos requisitos funcionales se encuentran desglosados dentro de los apartados siguientes: Modo historia y Crea tu propio análisis.

La decisión de indicarlos en este apartado no es otra que la de mostrar al lector, que estas dos funcionalidades forman parte de un todo y que no son dos soluciones inconexas pertenecientes a dos herramientas distintas, quedando, por tanto, la navegación entre funcionalidades como muestra la figura 3.1.

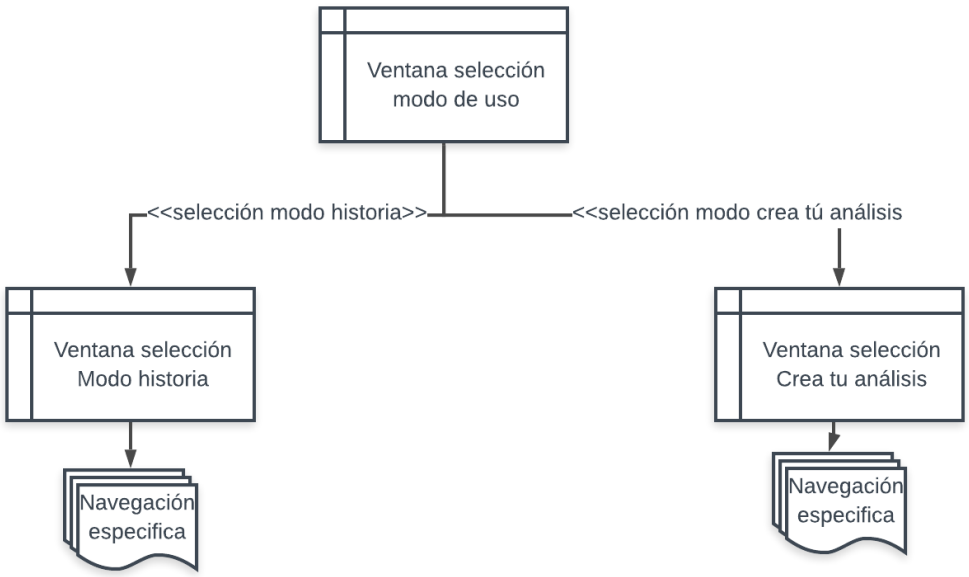

Figura 3.1: Diagrama de navegación inicial.

De igual forma, el diagrama de componentes que representa la solución propuesta donde se incluyen la utilización de herramientas externas queda detallado en la Figura 3.2

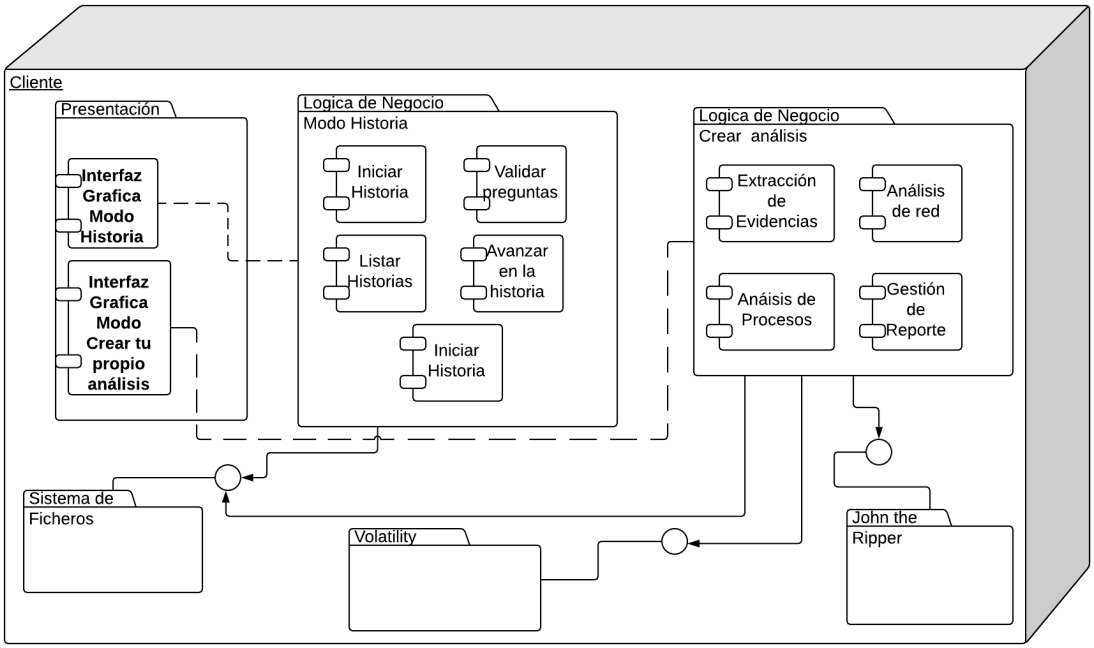

Figura 3.2: Diagrama de componentes.

3.2. Modo historia

Tal y como se comenta en la sección 3.1, el modo historia se centra en introducir conocimientos de una forma amena, para capturar el interés del alumno. Con este fin se ha optado por la posibilidad de incluir historias.
Una historia en este contexto es un incidente digital ocurrido y resuelto mediante la utilización del análisis de memoria volátil. Esta historia deberá contar con una descripción que permita al alumno situarse en el contexto del problema y que despierte su interés, ya que será fundamental en el periodo de aprendizaje de un tema tan amplio y complejo como es el análisis de memoria volátil. Además, de cara a que la solución sea extensible y a permitir que un docente pueda adaptar la herramienta a sus necesidades, Vol-E debe permitir añadir "historias". Este requisito requiere no sólo incluir texto descriptivo sobre un suceso, sino facilitar los pasos para definir nuevas historias. Los siguientes apartados describen la descripción del comportamiento de Vol-E para el modo historia, desde donde se extraen los requisitos funcionales.

3.2.1. Descripción del comportamiento

El funcionamiento del modo historia desde el punto de vista del alumno se describe a continuación tomando como base el diagrama de flujo de la Figura 3.3.

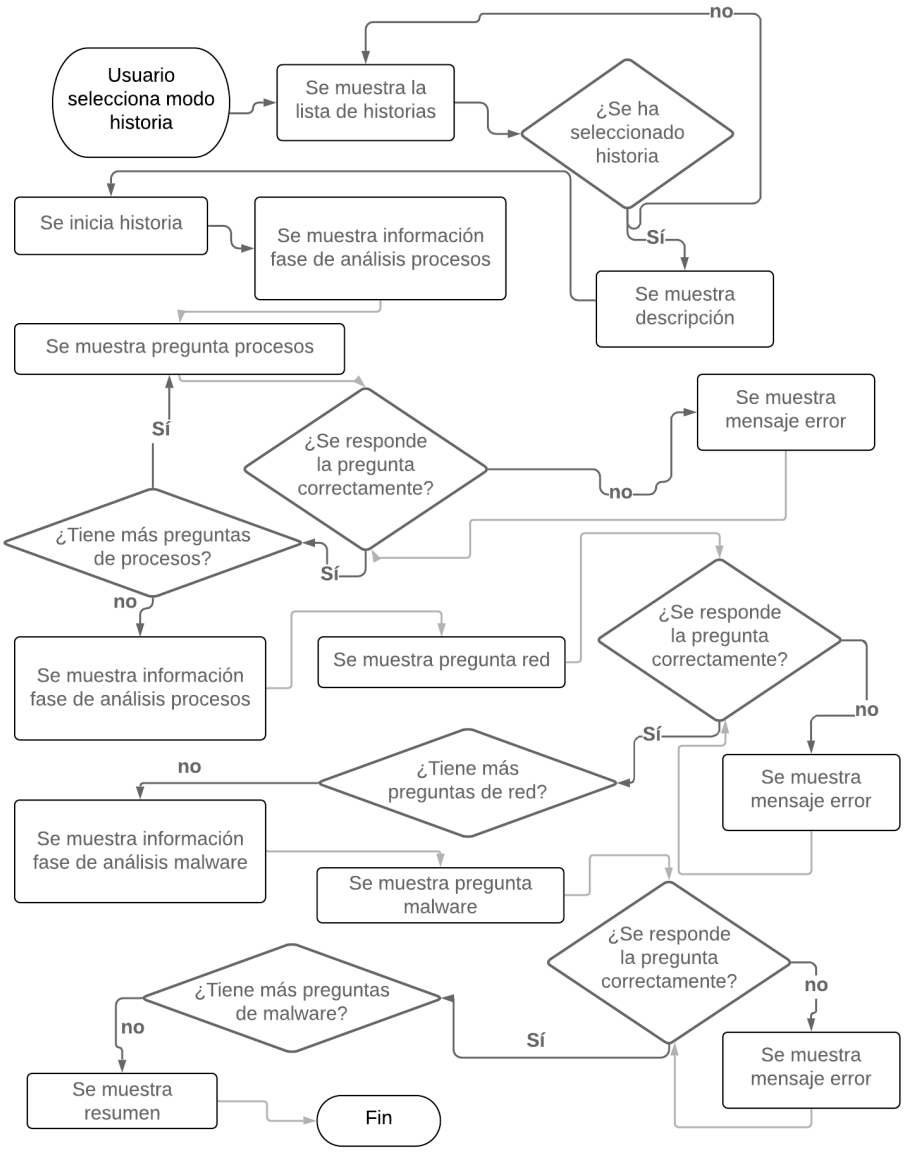

Figura 3.3: Diagrama de flujo modo historia.

Este flujo tiene como objetivo guiar al usuario hacia lo que serian los pasos básicos de un análisis de memoria volátil. Aunque en la realidad de un análisis de memoria volátil existe una gran interrelación entre los tres componentes, a fin de facilitar el

modo de aprendizaje se ha tomado esta decisión de diseño.
Tras seleccionar el modo historia, el alumno escogerá una de las historias disponibles, que contaría con una descripción básica. Entonces, el desarrollo de la historia se dividirá en 4 bloques principales: análisis de procesos, análisis de red, análisis malware y resumen.

La decisión de establecer esta división se basa, en que durante el análisis de memoria volátil de un incidente, las principales evidencias que pueden extraerse son o están relacionados con procesos. Como puede ser el caso, de los procesos que se encuentran en ejecución, que conjunto de drivers usa, si crea subprocesos, las DLLs que carga o sus permisos.

Una vez analizado los procesos, entra en escena la relación de estos con la red, si intentan establecer conexión internas o externas, los puertos abiertos, parámetros de configuración de red, DNS, búsquedas en red.

Y por último, ante un incidente siempre se busca saber el porqué, como ha ocurrido. Si existe algún elemento mal intencionado que ha producido el problema.

El apartado de resumen sirve para dar claridad a todas los conceptos que se han consolidado y enfocarlos en la historia. La unión del aprendizaje con un incidente o historia ocurrida permiten asentar los conocimientos, poniéndolos en practica bajo un entorno controlado, donde el alumno no se sentirá abrumado sin saber por donde empezar su búsqueda.

Por lo tanto, se le proveerá al alumno no de un conjunto de funcionalidades a ejecutar, si no de la información ya estructurada y dividida en los bloques anteriormente mencionados, para que este pueda distinguir que información es útil y cual no. Aprendiendo a diferenciar que elementos mostrados tienen relevancia para el incidente y cuales existen en el sistema pero no son fruto del incidente en cuestión.

La asimilación de conceptos se realiza a través de un conjunto de preguntas dentro de cada bloque. Estas preguntas deberán estar relacionadas con la información de cada bloque y relacionada con la historia en cuestión. Permitiendo al alumno responderlas y saber al instante si sus respuestas son o no correctas, Para ello y a fin de facilitar la evaluación de las preguntas, estas deberán contener 4 posibles respuestas, entre las que el usuario elegirá, validándose la veracidad de la misma.

El alumno estará obligado a responder las preguntas si desea transitar hacia el siguiente punto de la historia, permitiéndose de estar forma cerrar bloques de conocimiento, acotando el por tanto, conocimiento a fin de facilitar el aprendizaje.
Tal y como se comenta anteriormente, los procesos son la base más sencilla sobre la que parte el principio de un análisis de memoria volátil y desde este punto se parte hacia elementos de red, métodos de persistencias, actividades maliciosas, etc. Es por esto, por lo que la historia se cuenta de manera lineal, partiendo desde el bloque de

procesos, pasando por red y posteriormente al análisis malware, encontrando su fin en el resumen final de la historia.

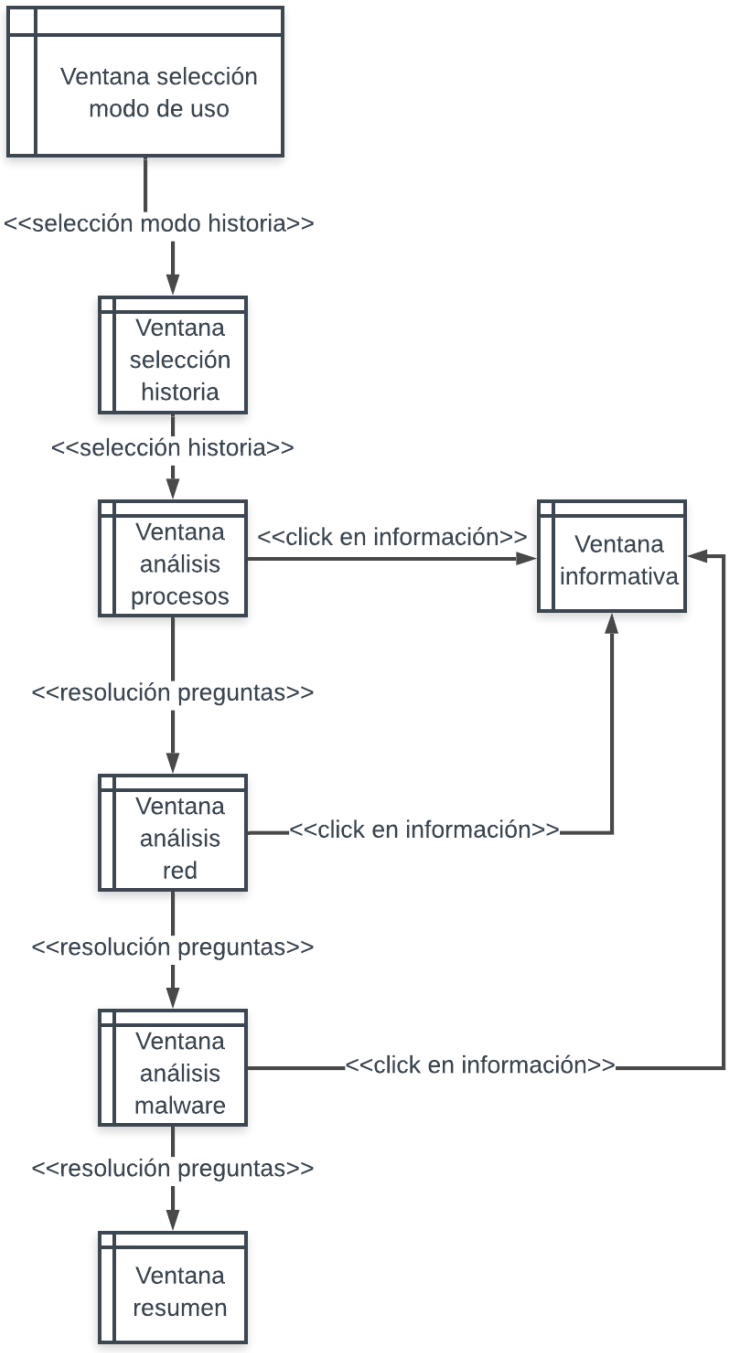

Figura 3.4: Diagrama de navegación modo historia.

Por último, a fin de fortalecer los conocimientos del alumno, cada bloque de la figura 3.3 cuenta con información acerca del propósito o formación sobre el área en cuestión, quedando por lo tanto el diagrama de navegación del modo historia tal y como muestra en la figura 3.4.

3.2.2. Definición y análisis de requisitos funcionales

Con toda la información anterior se pueden extraer los siguientes requisitos funcionales de la solución:

- RF1.1 - Añadir historias: La herramienta deberá permitirse añadir nuevas historias.
- RF1.2 - Listar historias: Permite al ver los historias disponibles.
- RF1.3 - Iniciar historia: Permite al alumno iniciar una historia mostrando toda la información relativa al análisis de procesos de la historia seleccionada, incluyendo preguntas.
- RF1.4 - Validar respuesta: Permite al alumno validar al usuario la respuesta seleccionada de una pregunta.
- RF1.5 - Avanzar en la historia: Permite al alumno avanzar en la historia, mostrando la información de la nueva fase.

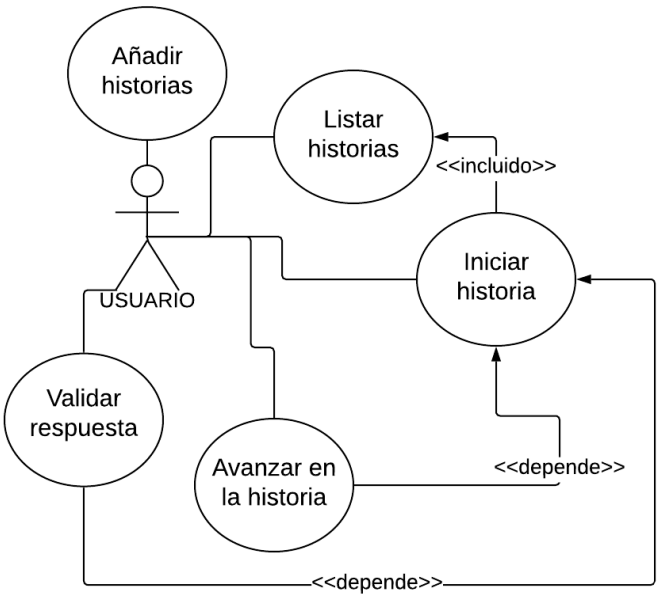

Figura 3.5: Diagrama caso de uso

Partiendo de los requisitos funcionales mencionados, se procede a realizar los casos de uso asociados a estos, dando como resultado el diagrama de casos de uso representado en la figura 3.5.

3.2.2.1. Añadir historias (RF1.1)

El requisito funcional RF1.1- Añadir historias, se ha decido diseñar no como una funcionalidad de la solución desde la interfaz gráfica, si no a través de la lectura de ficheros. Permitiendo de esta manera realizar cargas masivas de historias sin necesidad de cargarlas una a una.

Para añadir una historia sera por tanto necesario incluirla en el archivo de historias y añadir los 4 bloques anteriormente mencionados (análisis de procesos, análisis de red, análisis de malware y resumen) con sus correspondientes preguntas y soluciones.

3.2.2.2. Listar historias (RF1.2)

Este requisito funcional permite a un alumno obtener el conjunto de historias almacenadas.

La obtención de las historias, se realiza mediante la lectura de un archivo de texto desde el sistema de ficheros. Dicho archivo contiene por cada linea el nombre de una historia.

Una vez que el usuario selecciona una historia de la lista, que le ha sido mostrada a través de la interfaz gráfica, el sistema buscara una carpeta cuyo nombre sea el nombre de la historia seleccionada, encontrando en su interior, un archivo de texto con la descripción de la historia.

Pre-condición: El usuario debe haber seleccionado el modo historia al iniciar la herramienta. Debiéndose encontrar en la interfaz gráfica del modo historia.

Post-condición: Se deberá mostrar una lista de historias y la descripción de la historia seleccionada

Escenario principal: 1. El usuario selecciona el modo historia
2. El sistema busca la lista de historias, mostrándola al usuario.
3. El usuario selecciona una historia
4. El sistema busca la descripción de la historias seleccionada, la muestra y habilita el botón para iniciar historia.

Escenario alternativo 1. El usuario selecciona el modo historia
2. El sistema busca la lista de historias y sus descripciones. No existen historias en el fichero por lo que el sistema muestra un mensaje de error.

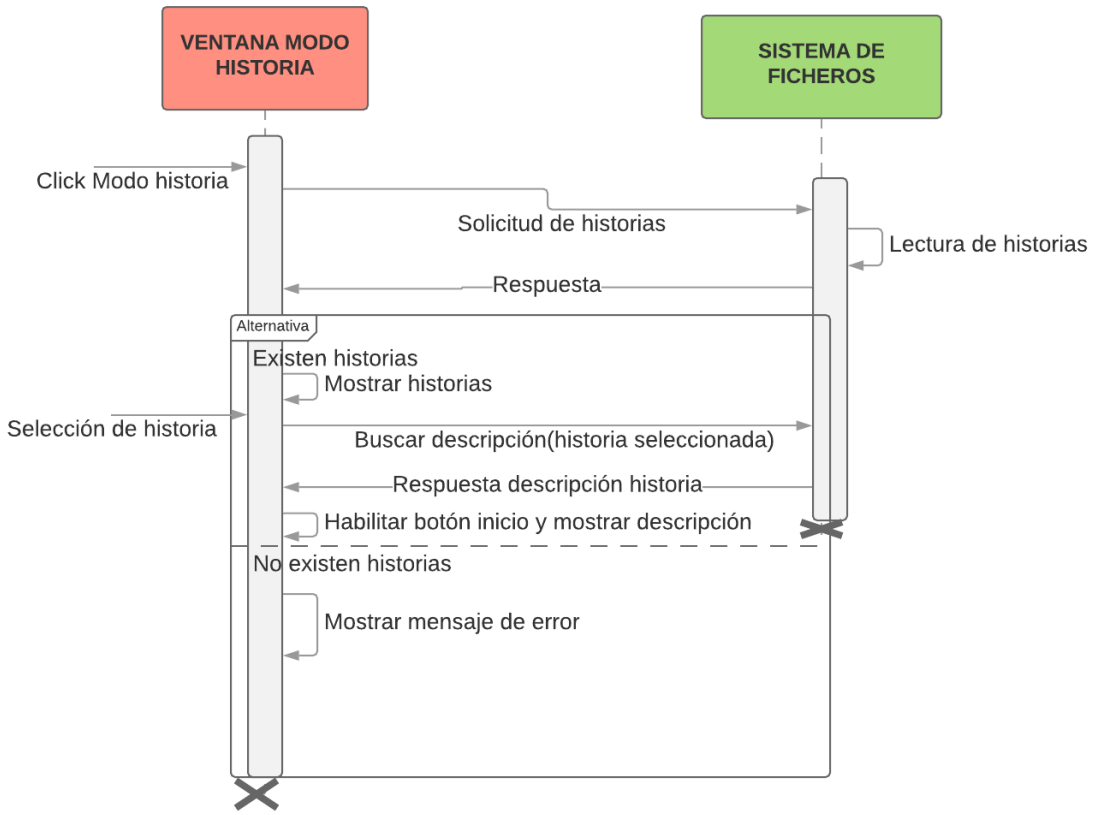

Figura 3.6: Diagrama de secuencia RF1.2 - Listar historias

En caso de que el fichero de texto que contiene la lista de historias haya sido borrado o no se encuentre en la ubicación correcta, el sistema mostrara al usuario un mensaje de error. Ocurriendo lo mismo en caso de no existir la carpeta de la historia seleccionada o el archivo de descripción.

3.2.2.3. Iniciar historia (RF1.3)

Este requisito funcional, permite a un alumno iniciar una historia. La Figura 3.7 muestra el diagrama de secuencia para esta acción.
Con el objetivo de iniciar la historia el alumno deberá haber listado las historias, habiendo seleccionado una de ellas. El análisis de proceso de una historia se inicia nada más pulsar el botón de historia, mostrándole al alumno la interfaz gráfica asociada y cargándose la información relativa a esta.

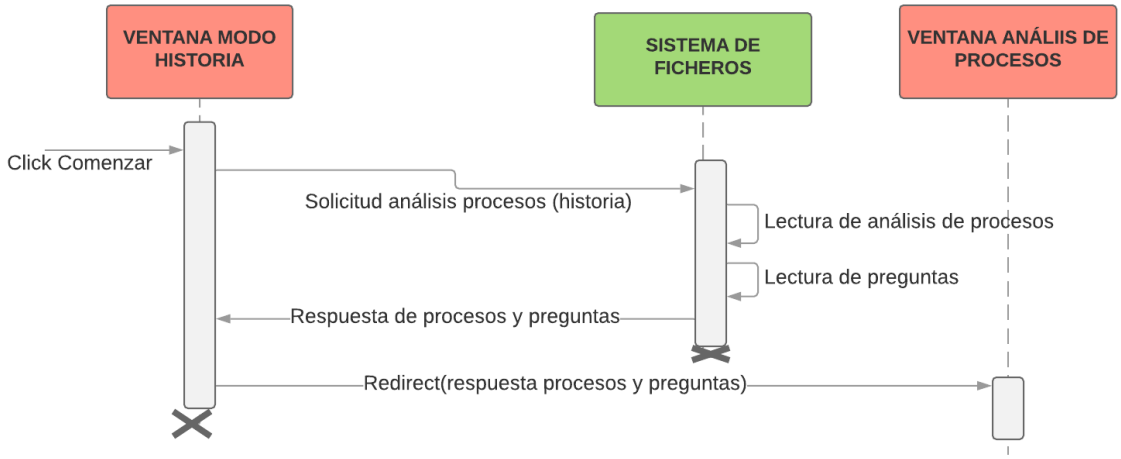

Figura 3.7: Diagrama de secuencia RF1.3 - Iniciar historia

El sistema buscara en el sistema de ficheros la carpeta con el nombre de la historia seleccionada, eligiendo de su interior la carpeta relativa al análisis de procesos. Obteniendo de esta ultima la información relativa al análisis de procesos y el conjunto de preguntas asociadas a esta fase del análisis.

Pre-condición: El usuario debe haber listado las historias.

Post-condición: Se deberá mostrar la lista de elementos de procesos, junto con las descripciones y preguntas asociadas.

Escenario principal: 1. El usuario selecciona una historia y pulsa iniciar.
2. El sistema busca la lista de elementos de procesos y preguntas en el sistema de fichero. Mostrando esta información al usuario.

Escenario alternativo En caso de que el sistema de ficheros no se capaz de encontrar la carpeta con el nombre de la historia seleccionada o no se encuentre en su interior la información relativa al análisis de procesos, el sistema mostrara al usuario la interfaz gráfica de la vista de análisis de procesos, mostrando únicamente la información encontrada.

3.2.2.4. Validar respuesta (RF1.4)

El sistema debe permitir validar la respuesta de una pregunta dentro del análisis de procesos, red y malware. La Figura 3.8 muestra el diagrama de secuencia para este requisito.

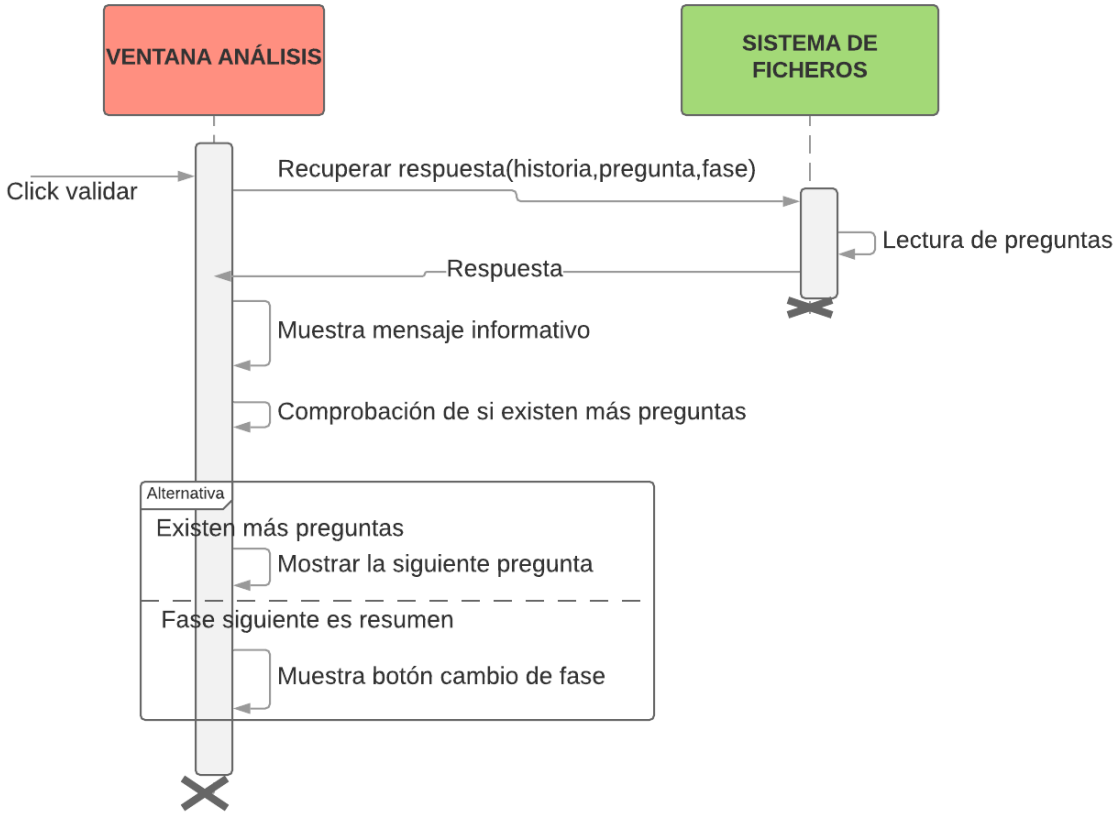

Figura 3.8: Diagrama de secuencia RF1.4 - Validar respuesta

Para la validación de la respuesta es necesario que un alumno pueda seleccionar una respuesta asociada a una pregunta. Dicha respuesta tiene asociado un identificador, el cual sera usado por el sistema para comprobar si el identificador de la respuesta correcta corresponde al identificador de la respuesta seleccionada por el usuario.

Pre-condición: El sistema debe encontrase dentro de la interfaz del análisis de procesos, red o malware.

Post-condición: Se mostrara un mensaje de respuesta confirmando si la respuesta es correcta o no.

Escenario principal: 1. El usuario selecciona una respuesta y pulsa el botón de comprobar.
2. El sistema comprueba cual es la respuesta correcta, mostrando mensaje informativo si ha sido correcta o no.
3. El sistema comprueba si existen más preguntas para el análisis en cuestión.
3.a En caso de encontrarse más preguntas, el sistema muestra la siguiente

pregunta al usuario.
3.b En caso de no encontrarse más preguntas, el sistema muestra el botón de cambio de análisis o resumen.

Escenario alternativo No contiene escenario alternativo.

Al igual que ocurre con los requisitos funcionales anteriores, el sistema no esta diseñado para validar el *JSON* de la estructura contenida en los ficheros que almacenan la historia, por lo tanto en caso no contener la estructura especificada el sistema mostrara la interfaz gráfica asociada a la fase correspondiente sin mostrar la información de dicha fase.

3.2.2.5. Avanzar en la historia (RF1.5)

A fin de transitar entre las distintas etapas con las que cuenta una historia, el sistema debe permitir avanzar en la historia una vez que las preguntas han sido respondidas para la fase actual. El diagrama de secuencia para este requisito se muestra en la figura 3.9.

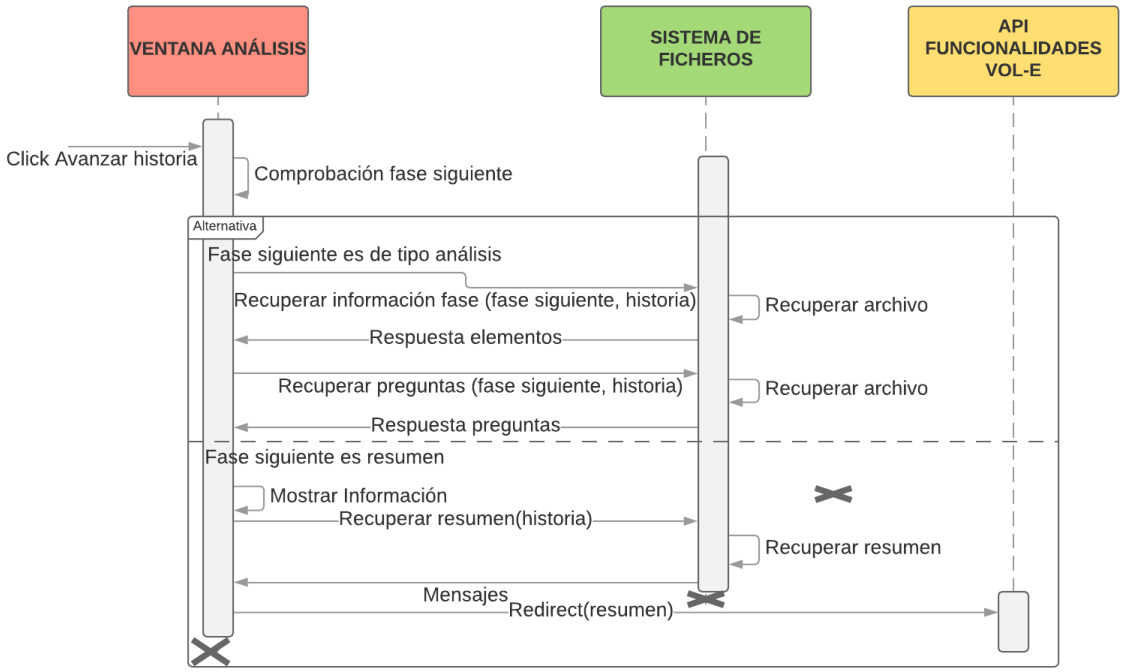

Figura 3.9: Diagrama de secuencia RF1.5 - Avanzar en la historia

Avanzar en la historia permite cambiar de fase en el análisis de memoria volátil, creando bloques estructurales diferenciados conforme se avanza en el análisis.

La solución propuesta esta diseña para solo permitir avanzar en la historia, imposibilitando la opción de volver a atrás en la misma. Esto se debe a que se pretende mostrar la historia como una secuencia de acontecimientos lineales, creando una mayor inmersión por parte del alumno, lo que mejora el aprendizaje de este.

Pre-condición: El alumno debe encontrase dentro de la interfaz del análisis de procesos, red o malware y deberá haber respondido a todas las preguntas y ejecutado su validación.

Post-condición: Se mostrara la nueva fase con respecto a la fase que se encontraba en la pre-condición.

Escenario principal: 1. El usuario hace click en el botón de cambio de fase correspondiente.
2. El sistema comprueba cual es la fase siguiente.
2.a La fase siguiente es una fase de análisis por lo que recoge del sistema de ficheros la información de los elementos de esa fase y sus preguntas correspondientes.
2.b La fase siguiente es una fase de resumen por lo que recoge del sistema de ficheros la información del resumen.
3. El sistema muestra la nueva información recuperada.

Escenario alternativo No contiene escenario alternativo.

Los casos de uso son una descripción de los pasos o actividades que deberán ser realizados por los actores para llevar a cabo un proceso. Estas actividades o pasos se realizan sobre la interfaz gráfica de usuario que se ha diseñado para esta solución. A fin de facilitar la posterior implementación, se han realizado un conjunto de maquetas de las interfaces que afectan al modo historia. Tal y como se pueden ver en las Figuras 3.10, 3.11, 3.12 y 3.13.

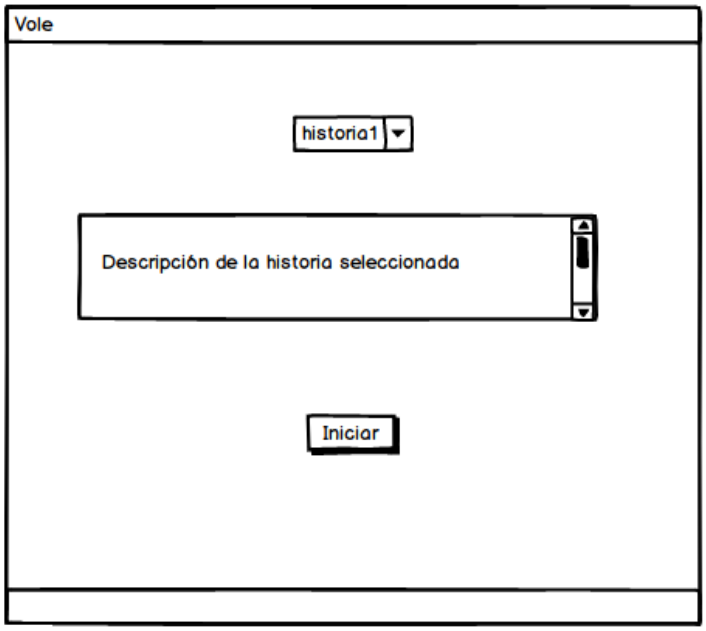

Figura 3.10: Maqueta ventana de selección de historia

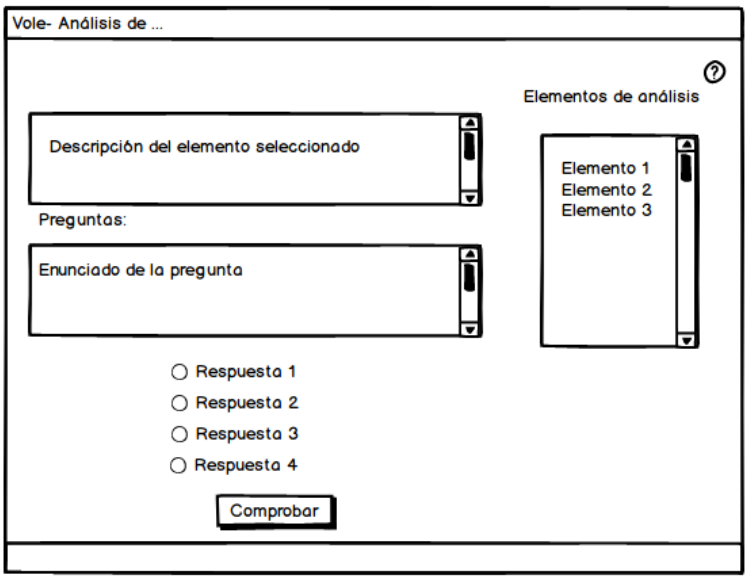

Figura 3.11: Figura maqueta ventana de análisis del modo historia

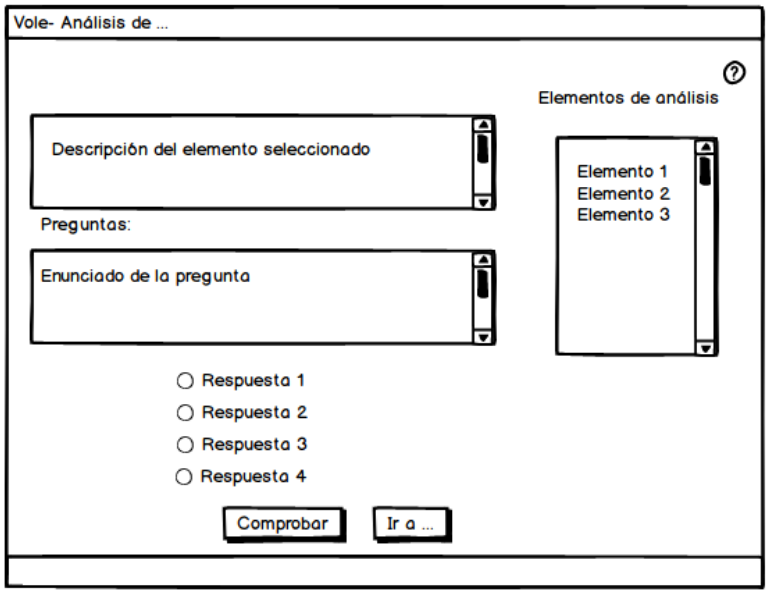

Figura 3.12: Maqueta ventana de análisis del modo historia permitiendo avanzar en la historia

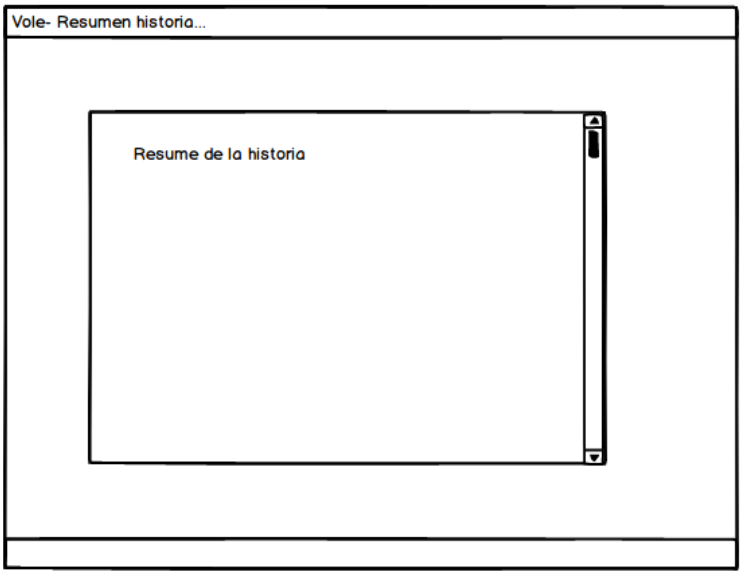

Figura 3.13: Maqueta ventana de resumen modo historia

3.3. Crea tu propio análisis

Una vez el alumno ha probado el modo historia para iniciarse en el contexto, adquiriendo las bases para el análisis de memoria volátil, es necesario proveerle de una herramienta que le facilite la inmersión durante sus primeros análisis de memoria volátil. Lo ideal es presentar una alternativa que permita realizar algunos análisis sobre la memoria pero sin usar directamente *Volatility*. Para cubrir estos objetivos se ha creado el modo *Crea tu propio análisis*.

Durante esta sección se aborda la definición de este modo, mediante una descripción general de los tipos de análisis considerados (procesos 3.3.2, red 3.3.3, extracción de evidencias 3.3.4) y la definición de dichos tipos para este modo. Además, se describirán los requisitos funcionales para este modo. A diferencia del modo anterior aquí las consultas se harán sobre la memoria seleccionada por el alumno.

3.3.1. Consideraciones previas sobre los tipos de análisis

El modo análisis al usuario de una herramienta con una fácil interfaz gráfica que le facilitara las principales funcionalidades para el inicio en el análisis de memoria volátil.

Estas funcionalidades están basadas en tres aspectos principales en el análisis de memoria volátil:

Procesos: Un proceso es una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistema asociados. Los procesos son el principal elemento sobre el que iniciar el análisis de memoria volátil y sobre estos hay un conjunto de recursos sobre los que es posible extraer información.

Elementos de red: Un elemento de red es todo aquel elemento que guarda relación con la red de un sistema, por ejemplo, socket, direcciones ips, puertos, firewall, etc.

Una vez analizados los procesos de un sistema, el paso más lógico en el análisis guiado de memoria volátil es analizar si estos procesos han realizado alguna conexión con el exterior, si ha habido elementos descargados o configuraciones de seguridad o red modificadas.

Extracción de evidencias: Asociado al análisis de procesos y elementos de red, adicionalmente es necesario extraer una serie de *items* (dlls, drivers, etc.) para ser analizados de manera individual a través de la utilización de herramientas de análisis de malware a fin de encontrar un uso ilegitimo.

Adicionalmente a los tres aspectos principales mencionados anteriormente, se proveerá al alumno de un reporte donde aparecerán de manera estructurada las funcionalidades que este ha realizado junto con sus resultados, a fin de ayudar al usuario a encontrar relación y guardar esta información para futuros análisis.

El análisis de la memoria realizada por esta herramienta, tomará como base a la herramienta *Volatility*, la cual contiene una gran cantidad de funcionalidades para el análisis de memoria volátil, siendo necesaria para la ejecución de estas funcionalidades la obtención del perfil de la memoria. Este perfil es el identificador del sistema operativo de la memoria volátil.

A fin de eliminar redundancias y mejorar los tiempos de ejecución, se realizara la obtención del perfil de la memoria de manera automática solo en una ocasión, en el momento en el que se seleccione la memoria que se desea analizar.
Recalculandose solo en caso de seleccionar otro archivo de memoria.

El modo *crea tu propio análisis* cuenta con cuatro interfaces gráficas diferenciadas, las cuales dividirán el análisis de los anteriores puntos mencionados. Ventana análisis de procesos, ventana análisis elementos de red, ventana extracción y ventana de reporte. De manera complementaria a estas cuatro interfaces principales, existirán dos interfaces, una para la selección del archivo de memoria que sera analizada y la otra para la selección de directorios donde se guardara el reporte y la extracción de evidencias.

El análisis de memoria volátil no es un proceso lineal si no que es necesario realizar saltos entre el análisis de procesos, la extracción de evidencias y el análisis de elementos de red. Es por esto por lo cual la herramienta desarrollada ha sido diseñada para permitir la navegación entre las diferentes interfaces gráficas de este modo de uso. La herramienta desarrollada ha sido diseñada además para almacenar y mostrar el resultado de la ultima funcionalidad ejecutada para cada tipo de análisis a fin de facilitar la navegación y ayudar al usuario a encontrar la información de manera clara y precisa.

El resultado de este diseño da como resultado el diagrama de navegación mostrado en la figura siguiente.

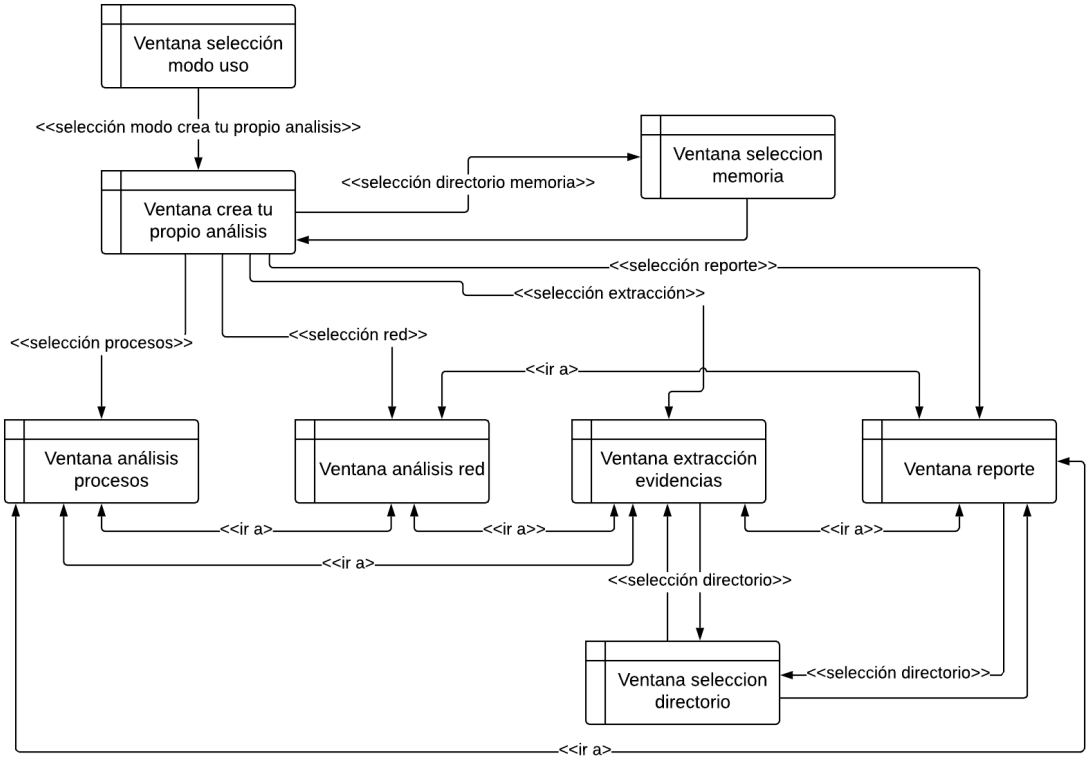

Figura 3.14: Diagrama de navegación en el modo *crea tu propio análisis*.

Para el análisis de memoria volátil es necesario conocer los elementos que rodean a los procesos, ya que estos son la base del análisis de memoria volátil.

Según se describe en el libro de referencia *The art of Memory Forensics*[1], los procesos están relacionados con los siguientes conceptos o elementos:

SIDs y privilegios Identificador del proceso y conjunto de privilegios asignados al mismo.

Tabla de manejadores: Contiene la tabla de manejadores de objetos del kernel relacionados con archivos, conexiones de red, semáforos, etc.

VADS: Son las regiones que utiliza Windows organiza la memoria asociadas a los procesos. Espacio de memoria virtual privado para cada proceso.

Módulos cargables: Conjunto de elementos que son cargados para la ejecución del proceso, como puede ser, ejecutables, dlls, drivers, pilas, etc.

Hebras: Hebras asociadas al proceso, encargadas de ejecutar las lineas de código.

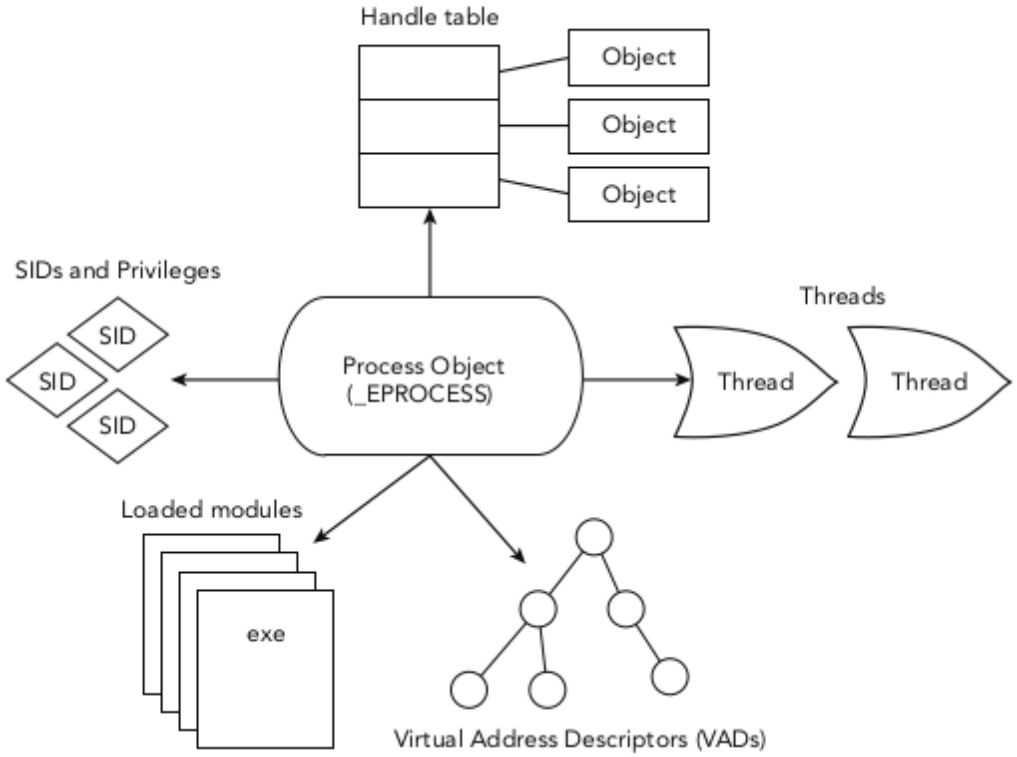

Figura 3.15: Diagrama de alto nivel de procesos del libro *The Art Of Memory Forensics* [1]

Sobre este diagrama se establecen las principales funcionalidades que incluye el diseño de la herramienta, dividida en tres bloques (procesos, red y extracción).

El resultado de la ejecución de las funcionalidades establecidas para el análisis de procesos y elementos de red tendrán dos vistas. Una vista resumida donde únicamente se mostrara la información principal y otra vista donde aparecerá toda la información relacionada con la ejecución de la funcionalidad seleccionada.

El objetivo de realizar estas dos vistas de resultados es ayudar al usuario no experto, focalizándolo y dirigiéndolo hacia la información principal, de modo que no se pierda ante una respuesta compleja.
De igual forma se le proveerá de la salida completa a fin de que pueda comprobar toda la información y que de esta forma vaya creciendo poco a poco en el proceso de aprendizaje.

3.3.2. Análisis de procesos

En primer lugar, el análisis de procesos debe incluir como no puede ser de otra forma, la funcionalidad para listar los procesos que se encontraban en ejecución en el momento en el que se realizo la extracción de la memoria volátil. Esta obtención de procesos se realiza recorriendo la lista doblemente vinculada a la que apunta PsActiveProcessHead que se encuentra en el *KDBG*(*kernel's process control block*).

La respuesta resumida de esta funcionalidad contendrá el nombre de los procesos que se estaban ejecutando así como el PID, que es el identificador de proceso a fin de que pueda servir como identificador para enlazar información con otras funcionalidades.

Por defecto, la lista de procesos que se están ejecutando en un sistema y que un alumno puede ver, no incluye los procesos ocultos.Siendo por tanto necesario implementar la funcionalidad que permita listar procesos ocultos.

Para la obtención de esta lista de procesos ocultos se realizara una consulta de los procesos desde 7 perspectivas distintas (listado de procesos, exploración de objetos de proceso, escaneo de subproceso, subprocesos de escritorio,procesos de sesión, tabla de manejador CSRSS y tabla de manejador PspCid) una vez obtenido el conjunto de procesos se compara con los procesos obtenidos en la primera funcionalidad implementada, quedando únicamente los que no se encuentran en la primera funcionalidad. Sobre este conjunto de procesos se devolverán al usuario los procesos cuyo tiempo de salida no este informado y tengan más de cero hebras. Esto indicara que los procesos se encuentran aún en ejecución.

Adicionalmente, en la respuesta resumida se mostraran los procesos que han modificado el tiempo de salida de manera fraudulenta. En ocasiones los atacantes del sistema pueden obtener privilegios y acceden al núcleo de memoria para sobrescribir el campo _EPROCESS.ExiTime del proceso, para que aparezca como que ha salido, de esta forma el proceso pasara desapercibido para el usuario.

Una manera de las maneras de detectarlo es debido a que los procesos que han acabado no tienen subprocesos y además tienen un identificador no válido. Por lo que, si se encuentran procesos que tienen tiempo de salida pero que contienen subprocesos asociados o un identificador válido, se puede establecer que es un proceso que ha sido modificado de manera fraudulenta.

A fin de facilitar la formación de los alumnos se ha creado otra funcionalidad que muestra los procesos críticos del sistema, estos procesos son procesos típicos de los sistemas operativos Windows que tienen una funcionalidad establecida y que deberá ser conocida por los usuarios.

La familiarización de los usuarios con estos procesos permitirá que realicen de manera sencilla un cribado y detecten anomalías en el sistema.

La lista de procesos críticos principales de los procesos Windows son : Idle, System, csrss.exe, services.exe, svchost.exe, lsass.exe, winlogon.exe, explorer.exe y smss.exe.

Tal y como se ha mostrado en el diagrama de alto nivel de procesos, uno de los elementos asociados a los procesos son las DLLs, por lo que la herramienta implementada incluye la funcionalidad para listar las DLLs asociadas a procesos.

Para obtener las DLLs se recorre la lista doblemente enlazada de estructuras *LDRDATATABLEENTRY* que se encuentra en el *InLadOrderModuleList* del *PEB*(*Process Environment Block*).Los procesos al cargar una biblioteca añaden a esta tabla la lista de DLLs usadas.

La respuesta resumida mostrara agrupadas por procesos las DLLs cargadas, el nombre de las mismas y el *path* de mapeo donde se encuentra. Mientras que la respuesta completa incluye adicionalmente el espacio de memoria donde se encuentra dicha DLL e indicando si se encuentra en memoria, cargada o inicializada.

De la misma forma que pasaba con los procesos, pueden existir DLLs ocultas, que o bien son cargadas de manera dinámica o bien han sido ocultadas de manera malintencionada modificando la tabla *PEB* con algún rootkit.

A fin de dar soporte al alumno en la búsqueda de estas DLLs, la herramienta ha sido diseñada para obtener las DLLs ocultas, para ello se comprueba de la lista de DLLs obtenida anteriormente si alguna no se encuentra en memoria, cargada o inicializada pero aún así tiene asociada un *path* del sistema para la DLL.

En otro apartado, la ejecución de procesos tiene asociado a cada uno de ellos una lista de privilegios, estos privilegios han podido ser asignados por el proceso padre que lo ha ejecutado o han sido auto establecidos por el propio proceso. La auto asignación de privilegios no indica per se que exista una intención mal intencionada del proceso. Pero si es un punto a tener en cuenta, donde el alumno deberá fijarse si quiere encontrar posibles anomalías.
Por ello, se ha establecido una funcionalidad que liste los privilegios auto-establecidos agrupados por servicios.

Al igual que pasaba con las DLLs, los procesos pueden tener asignadas llamadas a funcionalidades de drivers, a fin de que el usuario pueda conocer la lista de drivers que han sido cargados, se provee de la funcionalidad para obtener la lista de drivers y la tabla IRP asociado a estos, de esta manera se puede comprobar si existen drivers que son llamados a partir de otros drivers, pudiéndose comprobar si existe alguna llamada ilegitima a algún driver y este está utilizando métodos de ocultación.

En ocasiones los incidentes que provocan la necesidad de realizar un análisis de memoria volátil, son provocados por algún proceso malicioso, los cuales en muchos casos intenta persistir dentro del sistema.

Dos de los métodos de persistencias conocidos, consisten en la creación de determinadas claves de registro dentro de sistema o de la creación de servicios que son ejecutados al iniciar el sistema, volviendo a revivir el malware.

A fin de servir de guía y ayuda para el usuario se ha provisto a la herramienta con la funcionalidad de listar los registros de persistencia, de esta forma se devuelve al usuario los valores para las claves de registro de Windows asociadas a:

Microsoft\Windows\CurrentVersion\RunOncen

Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

Microsoft\Windows\CurrentVersion\Run

Microsoft\Windows NT\CurrentVersion\Windows

Microsoft\Windows NT\CurrentVersion\Windows\Run

Microsoft\Windows\CurrentVersion\Run

Microsoft\Windows\CurrentVersion\RunOnce

Estas son las claves de registro que tienen asociada persistencia dentro del sistema.

Por otro lado, se ha diseñado una funcionalidad para obtener los servicios con persistencia creados en el sistema, para ello en primer lugar se obtiene el *currentcontrolset* asociado al sistema para posterior acceder a los servicios asociados a dicha clave de registro.

La existencia de un ejecutable o servicio dentro de estas claves no significa que el proceso sea un malware, puede ser un proceso legitimo que necesita persistir en el sistema, pero puede servir como inicio para comenzar una investigación por parte del usuario de la herramienta.

Como punto y final en el ámbito del análisis de procesos, se puede recuperar el historial de comandos y consola que ha sido ejecutado en el sistemas anteriormente a la extracción de memoria volátil.

La obtención del historia de comandos y consola puede servir para comprobar que tipo de comandos han sido ejecutados en el *cmd* del sistema o si han sido ejecutados comandos en powershell o a través de una puerta trasera. Comprobándose si se ha realizado acciones de obtención de información o acciones malintencionadas.

La salida reducida contendrá únicamente los comandos ejecutados, mientras que la salida completa incluirá tanto los comandos como el resultado de ejecución de los mismos.

3.3.3. Análisis de red

En el análisis de red se pretende obtener la información relacionada con los elementos de la red, uno de estos elementos son las conexiones remotas existentes en el sistema. Para lo cual se recorre la lista de estructuras de conexión enlazadas individualmente a las que apunta un símbolo no exportado en el módulo *tcpip.sys*, en el cual se encuentran las conexiones *TCP* del sistema y que están activas en el momento de la adquisición.

Esta funcionalidad devolverá en la respuesta reducida la lista de conexiones, informando el proceso que está relacionado con la conexión, la dirección local y la dirección remota. En la respuesta completa se incluye la dirección de memoria donde se ha extraído esta información.

En ocasiones, es necesario conocer no solo las conexiones existentes, si no también las conexiones antiguas y que ya no están activas en el sistema. La obtención de conexiones antiguas se realiza mediante la búsqueda de *_TCPT_OBJECT* realizando un escaneo de etiquetas de grupo. La búsqueda a través de este método puede devolver información parcial, ya que es posible que parte de la información haya sido sobreescrita y no sea posible recuperarla.

Otro de los puntos donde se puede extraer información útil sobre el análisis de elementos de red, es la búsqueda de tarjetas de red en modo promiscuo. Las tarjetas de red en modo promiscuo se establecen para leer todo el contenido que circula por la red, esta es una formula muy utilizada para robar información de red muy útil para los atacantes.

Para ello, se realizara una funcionalidad que permite consultar el conjunto de socket vinculados al puerto 0 del protocolo 0 (HOPOPT) y un controlador abierto para \Dispositivo\RawIp\0. Devolviéndose la información del proceso asociado, la fecha de creación y la dirección de red para la salida reducida.

El análisis de elementos de red no solo incluye las conexiones, si no también el análisis de acceso a las URLs desde navegadores o la obtención del historial de navegación de Internet Explorer.

Para la obtención de las urls visitadas desde los navegadores, se realiza una búsqueda por fuerza bruta hacia el espacio de memoria perteneciente a los procesos asociados a los navegadores más utilizados actualmente, donde se busca cualquier cadena que contenga estructura de URL.

En el caso de la búsqueda del historial de Internet Explorer se recuperan los fragmentos de archivos de caché *index.dat* del historial de Internet Explorer a fin de encontrar URLs legibles que no hayan sido sobre escritas por el sistema.

Otro de los elementos críticos en el análisis de memoria volátil es el archivo que contiene las DNS, este archivo sirve como registro de las webs que se han visitado. El

navegador comprueba en primer lugar este archivo para intentar resolver el dominio y en caso de encontrarlo traduce el dominio por una dirección IP.

Este archivo es fundamental, ya que en caso de ser manipulado por algún malware puede redireccionar dominios a sitios no seguros o incluso cortar las actualizaciones de antivirus o firewalls.

La herramienta desarrollada está diseñada para obtener el archivo de cache de las DNS, recuperando así el archivo del sistema con nombre *hosts* y mostrando su contenido al usuario, para que este pueda encontrar incongruencias.

3.3.4. Extracción de evidencias

En numerosas ocasiones, el malware se esconde e inyecta código en las DLLs o drivers usado por procesos, es por tanto necesario permitir al alumno realizar una extracción de las DLLs y los drivers del sistema para posteriormente realizar un análisis de malware de estas.

La herramienta permite la extracción de todos los Drivers y DLLs cargados en el sistema, de forma que serán almacenados en la maquina del usuario para un posterior análisis.

Adicionalmente, se permite realizar la extracción de las DLLs cargadas por un proceso concreto o que se encuentren en un espacio de memoria especificado.

Por último, otro de los elementos que son habituales en la extracción de evidencias, son los hash de contraseñas. Estas son utilizadas para intentar acceder a las contraseñas y perfiles de los usuarios del sistema almacenadas en registro. Para lo cual, se busca en la estructura de claves de *SYSTEM* y *SAM*, una vez obtenido el hash se utiliza la herramienta de descentriptado por fuerza bruta *John the ripper*, lo cual permite en algunos casos mostrar al usuario la contraseña de dominio para los usuarios del sistema ya desencriptada.

Tal y como se ha comentado al principio de este apartado, toda la información obtenida de la ejecución de las funcionalidades por parte del usuario es necesario guardarla en forma de reporte, para posteriormente mostrarla al usuario permitiéndole a este almacenarla en un fichero.

Esta información debe estar agrupada por tipo de análisis y separando la información obtenida en la respuesta de resumen de la respuesta total de las funcionalidades.

Como se comenta anteriormente, el análisis de memoria volátil, no es un proceso lineal, por lo cual es difícil establecer un diagrama de flujo que cuadre con las posibles opciones que tiene el alumno.
Por lo tanto, se procede a mostrar uno de los diagramas de flujo más habilítales en el uso de este modo de la herramienta diseñada.

Donde se muestra la necesidad de obtener un perfil válido para el archivo de memoria seleccionado para el análisis.

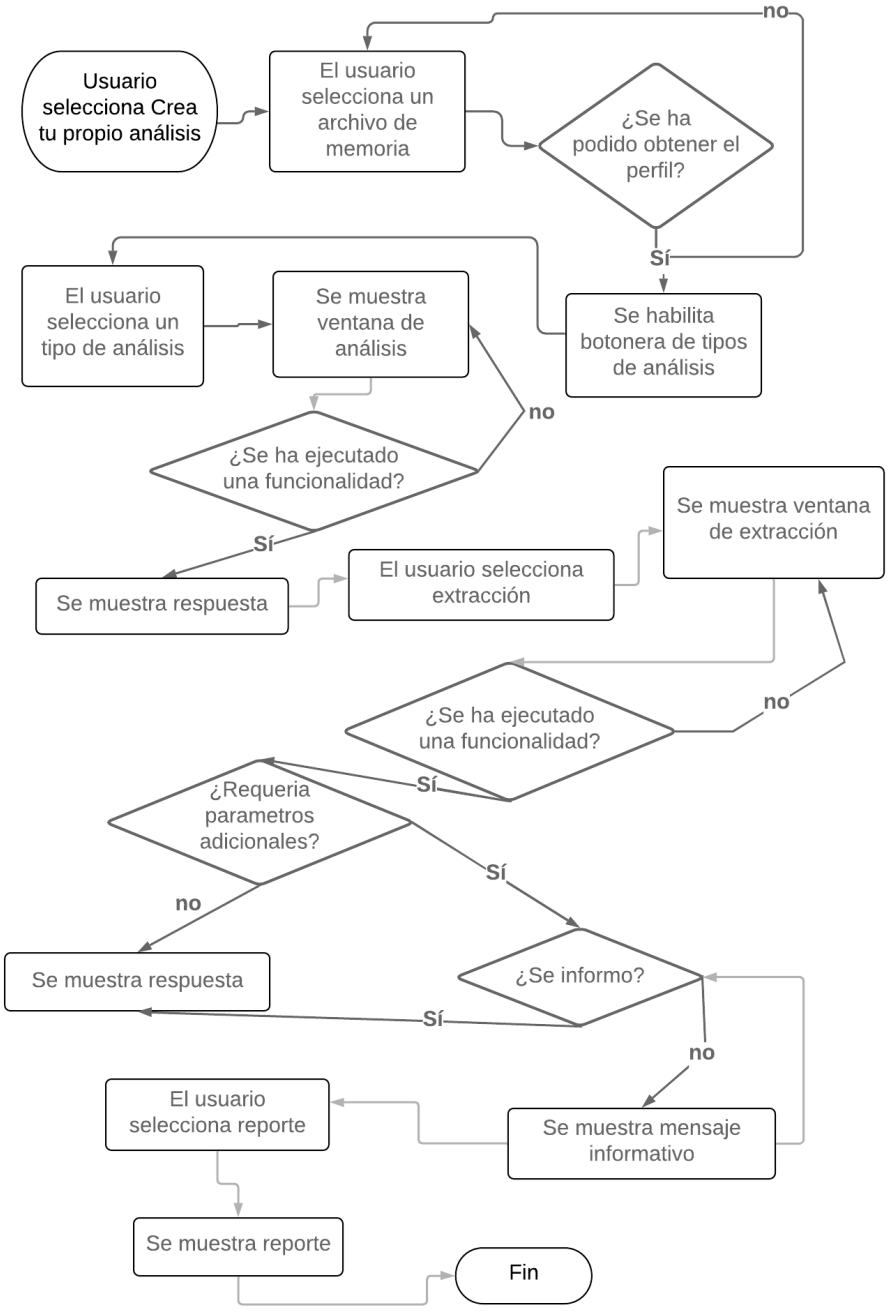

Figura 3.16: Diagrama de flujo Crea tu propio análisis.

3.3.5. Definición y análisis de requisitos funcionales

Con toda la información anterior se pueden extraer los siguientes requisitos funcionales de la solución:

- RF2.1 - Obtener perfil de memoria: Permite al usuario obtener el perfil del sistema asociado a la memoria seleccionada por el usuario
- RF2.2 - Analizar procesos: Permite al usuario realizar el análisis de procesos de una memoria seleccionada, lo cual incluye la ejecución de cualquiera de las funcionalidades anteriormente mencionadas en sección 3.3.2.
- RF2.3 - Analizar elementos de red: Permite al usuario realizar el análisis de red de una memoria seleccionada, lo cual incluye la ejecución de cualquiera de las funcionalidades anteriormente mencionadas en la sección 3.3.3.
- RF2.4 - Extraer Evidencias: Permite al usuario extraer evidencias de una memoria seleccionada, lo cual incluye la ejecución de cualquiera de las funcionalidades anteriormente mencionadas en la sección 3.3.4.
- RF2.5 - Generar Reporte: Permite al usuario revisar el reporte de las funcionalidades ejecutadas en los análisis de procesos y/o red de la memoria seleccionada.
- RF2.6 - Guardar Reporte: Permite al usuario guardar el reporte de las funcionalidades ejecutadas en los análisis de procesos y/o red de la memoria seleccionada.

Partiendo de los requisitos funcionales mencionados, se procede a realizar los casos de uso asociados a estos, dando como resultado el diagrama de casos de uso representado en la Figura 3.17.

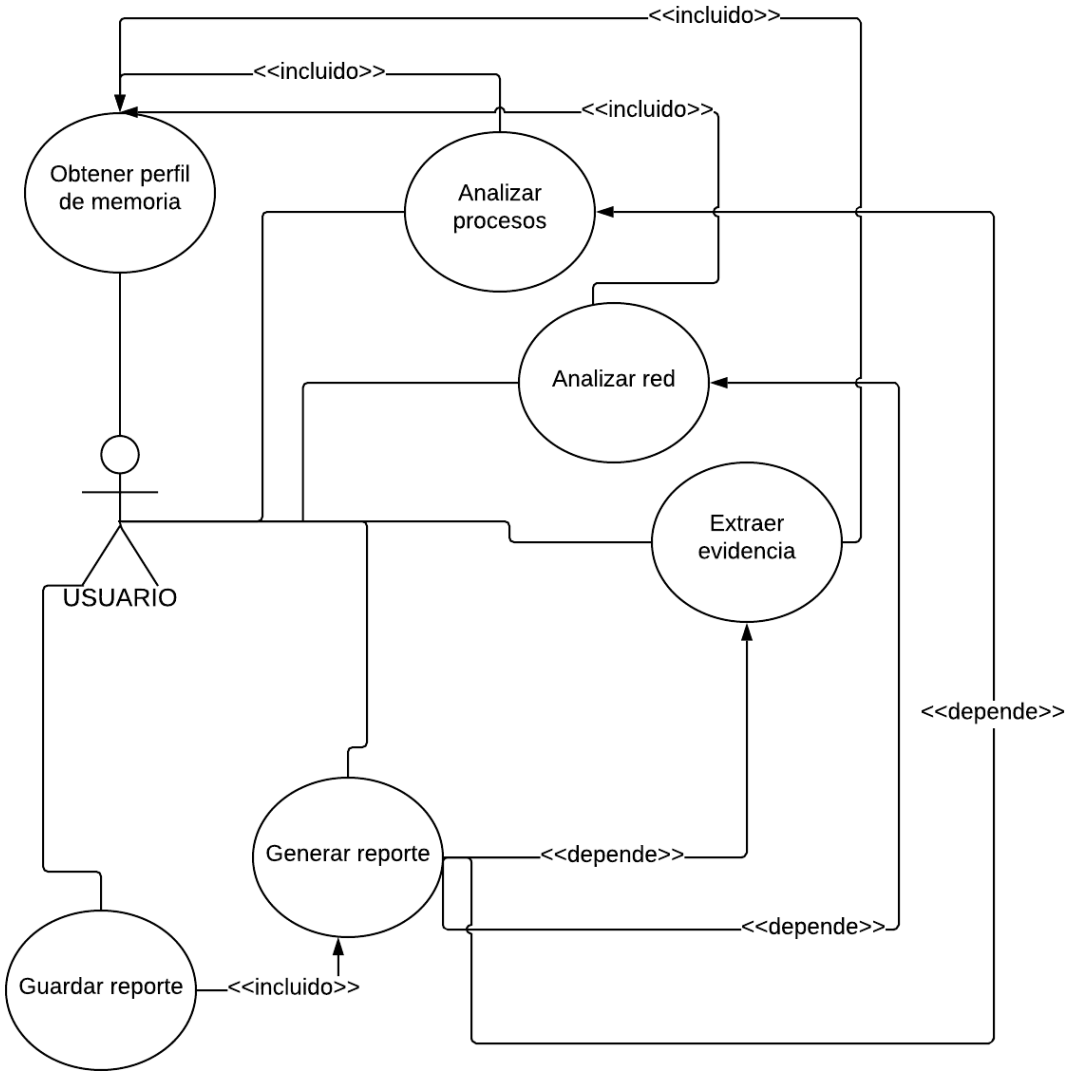

Figura 3.17: Diagrama caso de uso *Crea tu propio análisis*

3.3.5.1. Obtener perfil de memoria (RF2.1)

Permite al alumno obtener el perfil del sistema asociado a la memoria seleccionada por el usuario

Pre-condición: El alumno debe haber seleccionado el modo crea tu propio análisis al iniciar la herramienta y haber seleccionado un archivo de memoria.

Post-condición: Se deberá habilitar las opciones para realizar los análisis de procesos, red, extracción y reporte.

Escenario principal: 1. El usuario selecciona un archivo de memoria.
2. El sistema realiza una llamada a la api de funcionalidades implementadas de Vol-E, llamado estas a api de *Volatility*
3. La api de *Volatility* devuelve un perfil válido y el sistema lo almacena.
4. El sistema habilita la botonera para el inicio de las funcionalidades de análisis de proceso, red, extracción de evidencias y reporte.

Escenario alternativo 1. La api de Vol-E no es capaz de obtener un perfil válido.
2. El sistema muestra mensaje de error al usuario.

Quedando el diagrama de secuencia tal como se muestra en la figura siguiente.

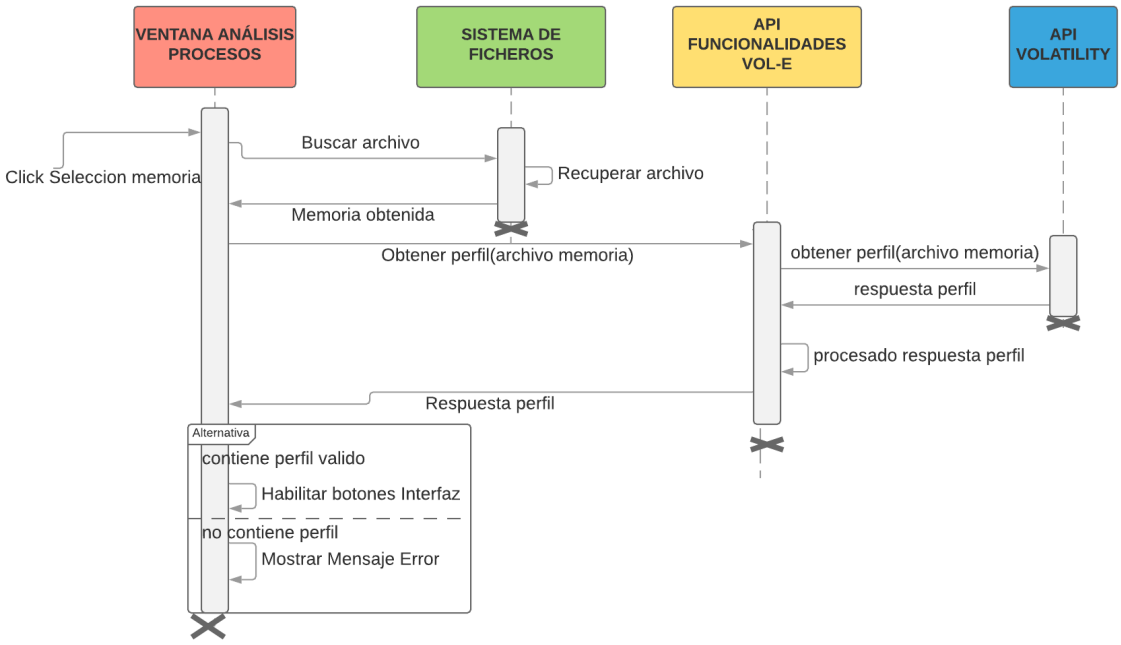

Figura 3.18: Diagrama de secuencia RF2.1 - Obtener perfil de memoria

RF2.2 - Analizar procesos

Permite al alumno realizar el análisis de procesos de una memoria seleccionada, lo cual incluye la ejecución de cualquiera de las funcionalidades anteriormente mencionadas en la sección 3.3.2.

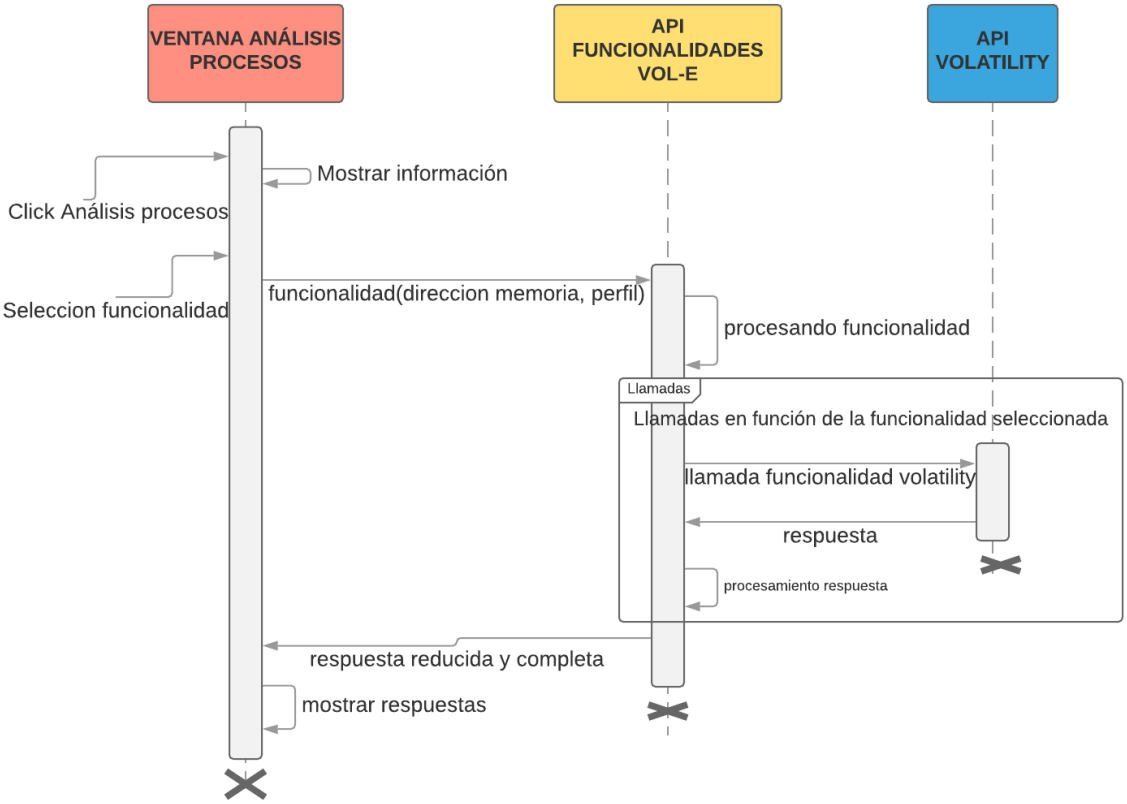

Figura 3.19: Diagrama de secuencia RF2.2 - Analizar procesos

Pre-condición: El alumno debe haber seleccionado el modo crea tu propio análisis al iniciar la herramienta y haber seleccionado un archivo de memoria de la cual se ha podido obtener un perfil válido.

Post-condición: El sistema mostrara el resultado de la funcionalidad seleccionada

Escenario principal: 1. El usuario seleccionara análisis de procesos.
2. El sistema mostrara la ventana de análisis de procesos.
3. El usuario seleccionara una funcionalidad de la lista y ejecutara.
4. El sistema realizara la llamada a la funcionalidad implementada en Vol-E api y esta llamara a la api de *Volatility*, procesando la respuesta.
5. El sistema mostrara la información de respuesta.

Escenario alternativo No se contempla

3.3.5.2. Analizar elementos de red (RF2.3)

Permite al usuario realizar el análisis de red de una memoria seleccionada, lo cual incluye la ejecución de cualquiera de las funcionalidades anteriormente mencionadas en la sección 3.3.3.

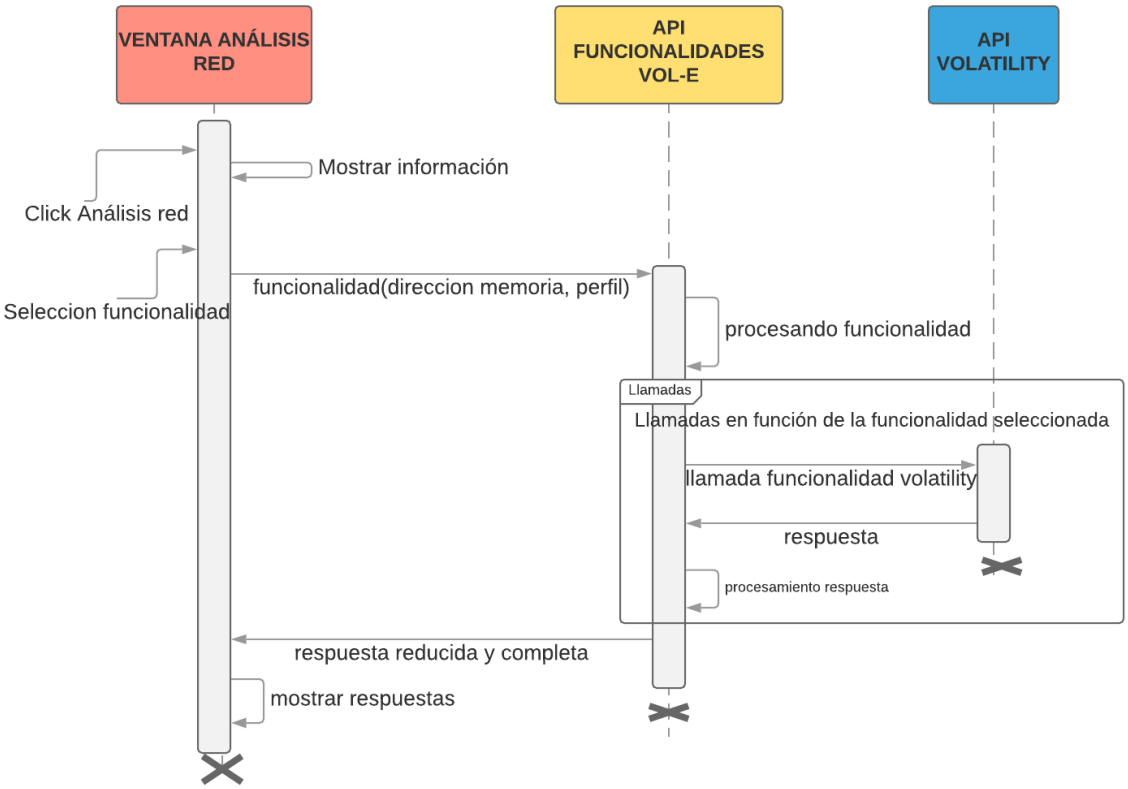

Figura 3.20: Diagrama de secuencia RF2.3 - Analizar elementos de red

Pre-condición: El usuario debe haber seleccionado el modo crea tu propio análisis al iniciar la herramienta y haber seleccionado un archivo de memoria de la cual se ha podido obtener un perfil válido.

Post-condición: El sistema mostrara el resultado de la funcionalidad seleccionada

Escenario principal: 1. El usuario seleccionara análisis de red.
2. El sistema mostrara la ventana de análisis de red.
3. El usuario seleccionara una funcionalidad de la lista y ejecutara.
4. El sistema realizara la llamada a la funcionalidad implementada en Vol-E

api y esta llamara a la api de volatility, procesando la respuesta.
5. El sistema mostrara la información de respuesta.

Escenario alternativo No se contempla

3.3.5.3. Extraer Evidencias (RF2.4)

Permite al alumno extraer evidencias de una memoria seleccionada, lo cual incluye la ejecución de cualquiera de las funcionalidades anteriormente mencionadas en la sección 3.3.4.

Pre-condición: El alumno debe haber seleccionado el modo crea tu propio análisis al iniciar la herramienta y haber seleccionado un archivo de memoria de la cual se ha podido obtener un perfil válido, habiendo seleccionado la ventana de extracción.

Post-condición: El sistema habrá creado un archivo de texto con el archivo del reporte

Escenario principal: 1. El usuario hace selecciona una funcionalidad .
2. El sistema comprueba si la funcionalidad requiere parámetro.
. 3.a La funcionalidad requiere parámetro, el sistema habilita el campo para introducir el parámetro
4.a El usuario inserta el parámetro y pulsa el botón de ejecutar.
5. El sistema mostrara la ventana para que el usuario seleccione directorio donde guardar reporte.
6. El usuario seleccionara un directorio y nombre de archivo.
7 El sistema ejecuta la funcionalidad de la api implementada para Vol-E, realizando las llamadas necesarias hacia la api de *Volatility* extraer la evidencia.
8 El sistema muestra la información relacionada con la extracción.

Escenario alternativo 3.b La funcionalidad no requiere parámetro.

Quedando el diagrama de secuencia tal como se muestra en la figura siguiente.

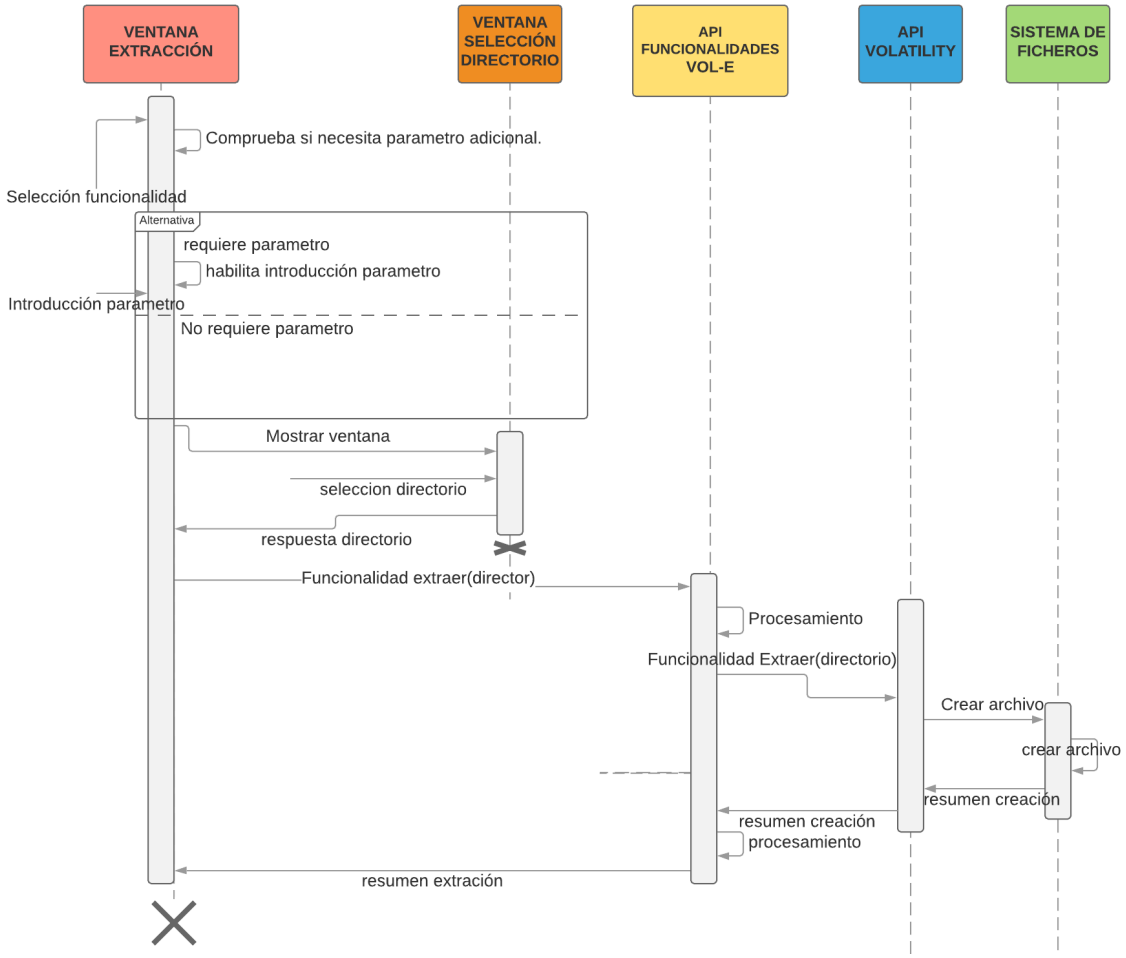

Figura 3.21: Diagrama de secuencia RF2.4 - Extraer Evidencias

3.3.5.4. Generar Reporte (RF2.5)

Permite al usuario revisar el reporte de las funcionalidades ejecutadas en los análisis de procesos y/o red de la memoria seleccionada.

Pre-condición: El usuario debe haber seleccionado el modo crea tu propio análisis al iniciar la herramienta y haber seleccionado un archivo de memoria de la cual se ha podido obtener un perfil válido.

Post-condición: El sistema mostrara el reporte generado con las funcionalidades anteriormente consultadas.

Escenario principal: 1. El usuario seleccionara reporte.
2. El sistema consulta la información almacenada sobre las funcionalidades

consultadas sobre la memoria.
3. El sistema muestra la información estructurada sobre la información almacenada.

Escenario alternativo No se contempla

Quedando el diagrama de secuencia tal como se muestra en la figura siguiente.

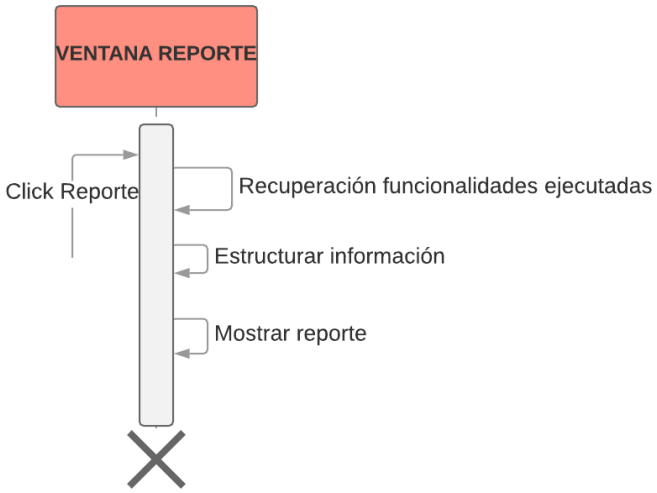

Figura 3.22: Diagrama de secuencia RF2.5 - Generar Reporte

3.3.5.5. Guardar Reporte (RF2.6)

Permite al usuario guardar el reporte de las funcionalidades ejecutadas en los análisis de procesos y/o red de la memoria seleccionada.

Pre-condición: El usuario debe haber seleccionado el modo crea tu propio análisis al iniciar la herramienta y haber seleccionado un archivo de memoria de la cual se ha podido obtener un perfil válido, habiendo seleccionado la ventana de reporte.

Post-condición: El sistema habrá creado un archivo de texto con el archivo del reporte

Escenario principal: 1. El usuario hace click en guardar,
2. El sistema mostrara la ventana para que el usuario seleccione directorio donde guardar reporte,
3. El usuario seleccionara un directorio y nombre de archivo,
4. El sistema recupera el reporte y genera el archivo.

Quedando el diagrama de secuencia tal como se muestra en la figura siguiente.

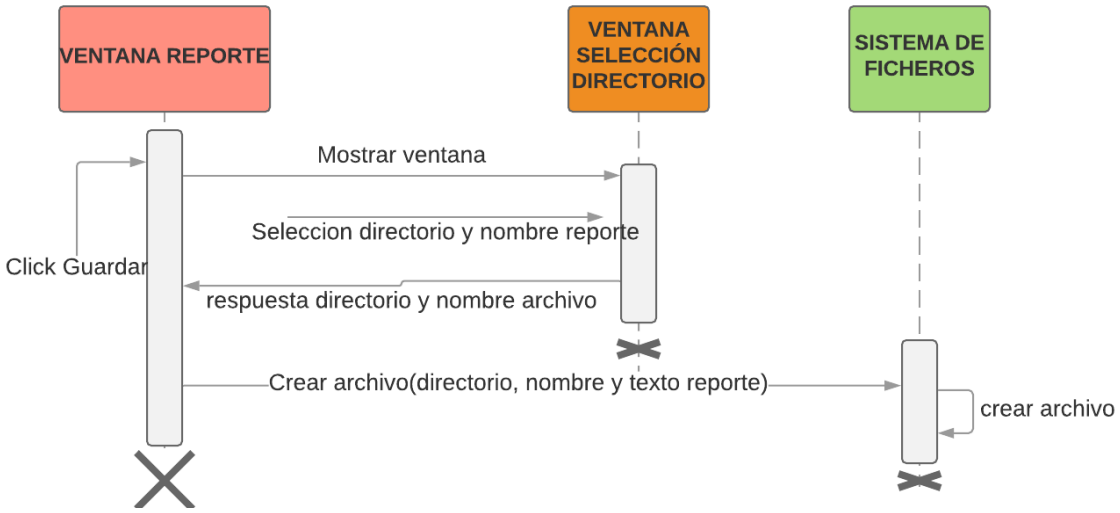

Figura 3.23: Diagrama de secuencia RF2.6 - Guardar Reporte

A fin de facilitar la posterior implementación, se han realizado un conjunto de maquetas de las interfaces que afectan al modo crea tu propio análisis.

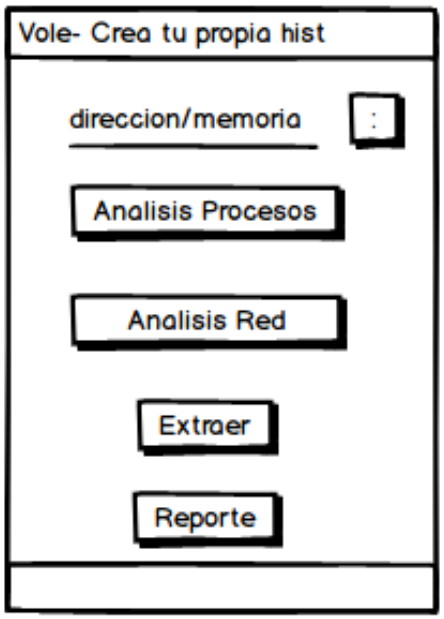

Figura 3.24: Maqueta ventana de selección memoria

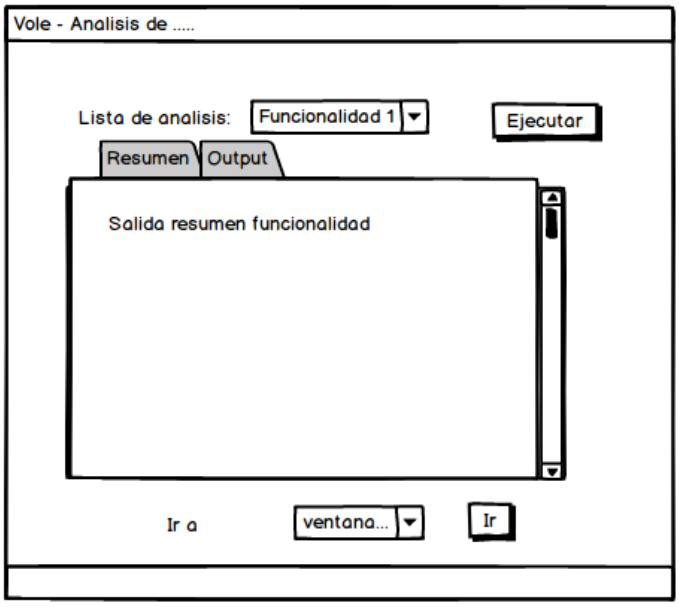

Figura 3.25: Figura maqueta ventana de análisis de procesos y red

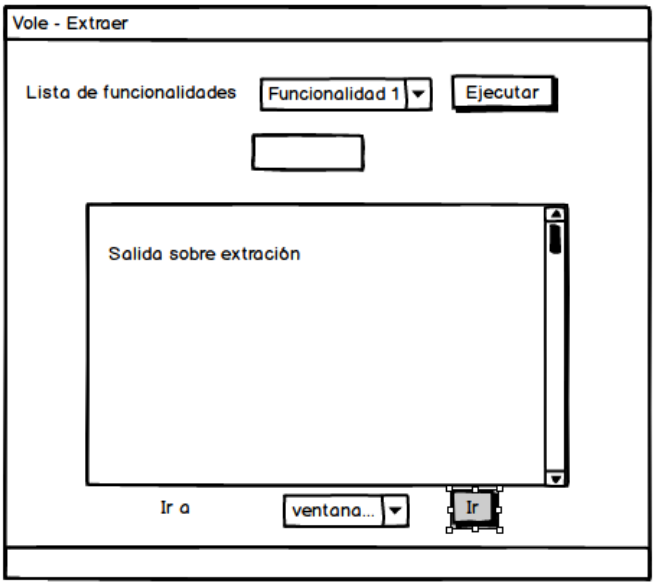

Figura 3.26: Maqueta ventana de extracción de evidencias

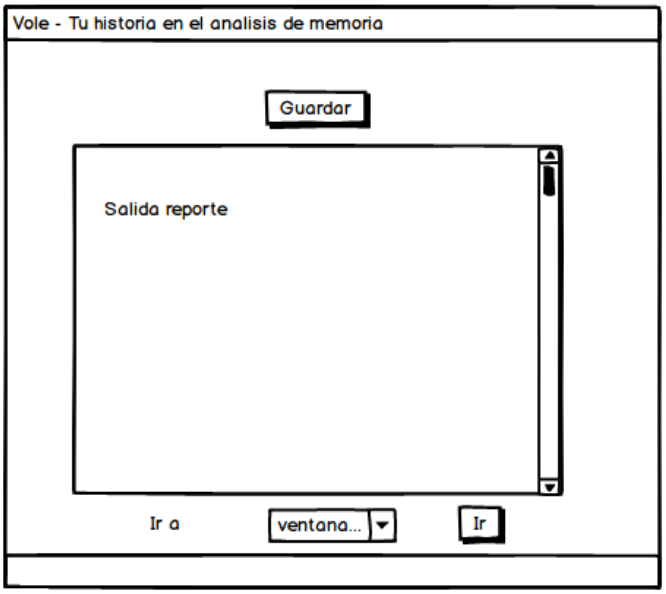

Figura 3.27: Maqueta ventana de reporte

Capítulo 4

Desarrollo de la solución propuesta

Una vez que se ha realizado el análisis de las soluciones existentes y el diseño de la solución propuesta, se procede a realizar el desarrollo de la herramienta Vol-E, que tal y como se comenta en apartados anteriores se implementa en *Python 3.7* y *Tkinter* para el desarrollo de la interfaz gráfica de usuario.

La solución propuesta contiene dos archivos .py, que son *vole.py* y *vole_ api_ expansion.py*. El archivo *vole.py* es el encargado de iniciar la aplicación desarrollada en *Python*. La ejecución de dicho archivo inicia la interfaz gráfica de inicio, la cual es una ventana donde se le permite al usuario elegir el modo de uso.

Esta ventana principal de la aplicación se ha creado gracias al paquete *Tkinter* y contiene dos botones, permitiendo elegir una de las dos opciones, tal y como aparece en la figura 4.1.

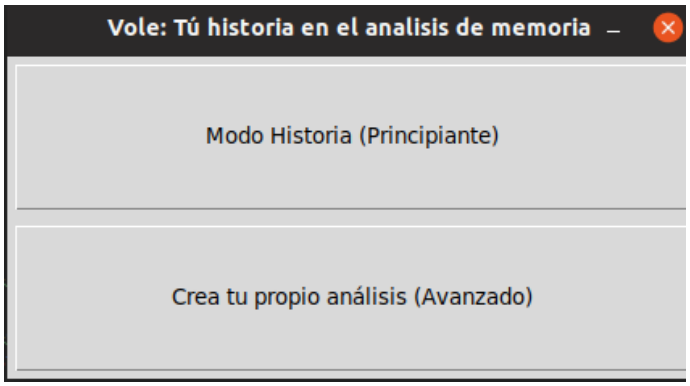

Figura 4.1: Ventana principal

Dentro del archivo *vole_ api_ expansion.py* se encuentran la funcionalidades de análisis de memoria volátil que han sido desarrolladas y que serán usadas dentro del modo Crea tu propio análisis.

4.1. Interfaces para el *Modo historia*

A fin de cumplir con los requisitos funcionales diseñados en el la sección 3.2, se han desarrollado tres interfaces gráficas distintas, cuya utilización se destaca a continuación.

4.1.1. Configuración de historias

La primera de ellas es la encargada de listar las historias obtenidas y mostrar la descripción de la historia seleccionada por el usuario.

Para ello en primer lugar Vol-E accederá a un listado de historias disponible, que podría ser ampliado por el docente. Se ha definido para esta causa un archivo de texto llamado *listahistorias.txt*, este archivo debe encontrarse dentro de la carpeta *historias* la cual debe encontrarse en el mismo nivel que el archivo desarrollado *vole.py*.
Dentro de este archivo se encontrara los nombres de las historias, cada nombre deberá encontrarse en una linea distinta del archivo de texto, siendo por tanto el separador entre ellas el retorno de carro.

El conjunto de nombres de historias obtenidas en la consulta anterior sera mostrada dentro de un cuadro de selección.

Una vez que el usuario ha seleccionado una historia, se procede a realizar una búsqueda de la descripción de dicha historia. Para ello, se busca una carpeta dentro de la carpeta *historias*, cuyo nombre ha de ser igual al nombre de la historia seleccionada. Dentro de esta carpeta debe encontrarse un archivo de texto llamado *descripcion.txt* desde donde se recuperara y mostrara al usuario la información contenida en esta. Dicha información no requiera seguir ninguna estructura, siendo texto libre.

En caso de no encontrarse dicha carpeta o no encontrarse el archivo *descripcion.txt* se le mostrara al usuario un mensaje de error. Este error indicara que no se creo la historia de manera correcta o que esta ha sido borrada del sistema de ficheros.

A fin de controlar la selección por parte del usuario de una historia dentro del desplegable, se le asigna a este desplegable un procedimiento de obtención del valor seleccionado, limpiando adicionalmente el cuadro que contiene la descripción para eliminar una posible descripción anterior, llamando posteriormente al procedimiento encargado de la obtención de la descripción de dicha historia.

```
self.combobox.bind("<<ComboboxSelected>>", self.selection_changed)
    def selection_changed(self, event):
        self.textDescripcionHistoria.config(state="normal")
        self.textDescripcionHistoria.delete(1.0, END)
```

```
self.textDescripcionHistoria.update()
self.historiaSeleccionada = self.combobox.get()
if self.historiaSeleccionada is not None:
    self.obtener_descripcion_historia()
```

Quedando la interfaz gráfica de dicha ventana tal y como muestra la imagen 4.2.

Figura 4.2: Ventana selección historia

Adicionalmente, a fin de controlar que el usuario no intente comenzar una historia sin haber seleccionado una con posterioridad, se ha establecido un control por el cual si no se ha seleccionado historia se mostrara al usuario un mensaje informando que es necesario seleccionar una historia para poder comenzar. Este control se ha realizado comprobando si ha sido posible recuperar un nombre de historia dentro del desplegable.

Quedando por lo tanto cumplimentado el requisito funcional *RF1.2*.

El botón comenzar de la ventana creada anteriormente es el encargado de destruir la ventana actual y crear la próxima ventana. Esta nueva ventana sera la ventana de análisis en su fase de análisis de procesos tal y como se especifico en el diseño de la solución.

4.1.2. Listado de elementos de análisis

La nueva ventana sera la encargada de mostrar la lista de elementos relacionados con la fase en la que se encuentra el análisis junto con su información, la lista de preguntas y posibles respuesta y la información de ayuda de la fase en cuestión.
Cabe destacar que esta ventana sigue el diseño empleado para *Crea tu Propio Análisis* a fin de que el alumno se familiarice con el significado de los colores empleados, que diferenciarán los tipos de análisis.

Para ello en primero lugar se procederá a obtener la información de la lista de elementos y su descripción. De igual forma que se hizo anteriormente, es necesario acceder al sistema de ficheros donde dentro de la carpeta con el nombre de la historia, al mismo nivel que el archivo de *descripcion.txt* anteriormente mencionado, deben encontrarse 4 carpetas con los nombres *analisisProcesos*, *analisisRed*, *analsisMalware*, *resumen*. Según la fase del análisis donde se encuentre el alumno, el sistema accederá a una de ellas.

En el interior de las tres primeras carpetas mencionadas se deberá encontrar dos archivos con los nombres *preguntas.txt* y *listadoProcesos.txt* o *listadoRed.txt* o *listadoMalware.txt* dependiendo de la fase en la que nos encontremos y a la carpeta que se haya accedido.

El archivo *preguntas.txt* cuenta con la información de las preguntas, las posibles respuestas y la respuesta correcta. Dicho archivo deberá contener la siguiente estructura JSON:

```
[{
   "id": "valor del identificador e la pregunta,
   debe ser unico",
   "pregunta": "Descripcion de la pregunta",
   "opcion1": "Descripcion posible respuesta 1",
   "opcion2": "Descripcion posible respuesta 2",
   "opcion3": "Descripcion posible respuesta 3",
   "opcion4": "Descripcion posible respuesta 4",
   "respuestacorrecta": "valor del id de la
   pregunta correcta"}]
```

Esta estructura contiene un array de preguntas, de las cuales la ventana de análisis inicialmente mostrara la primera pregunta.

El archivo *listadoProcesos.txt*, *listadRed.txt* y *listadoMalware.txt* contienen la información de los elementos de análisis de la fase en cuestión.
Estos archivos contendrá la siguiente estructura JSON:

```
[{"proceso": "nombre elemento",
  "descripcion": "Informacion del elemento"}]


[{"red": "nombre elemento",
   "descripcion": "Informacion del elemento"}]

[{"malware": "nombre elemento",
  "descripcion": "Informacion del elemento"}]
```

La información obtenida de los ficheros en cuestión sera mostrada según la fase en la ventana anteriormente creada tal y como se muestra en la figura 4.3.

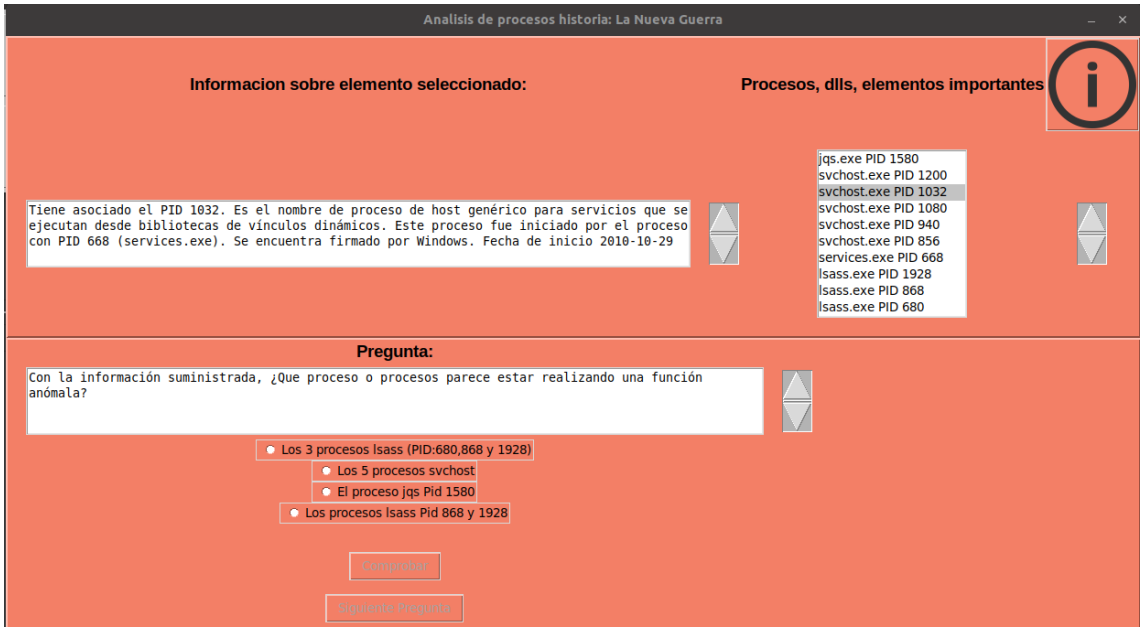

Figura 4.3: Ventana análisis

A fin de facilitar la asociación de conceptos, el color de la ventana de análisis cambiara en cada fase

El botón de ayuda que aporta información sobre la fase en la que se encuentra el análisis, creara una ventana emergente con la información relativa a la ayuda de dicha fase. La información de ayuda se encuentra *hardcodeada* en la herramienta para cada una de las fases.

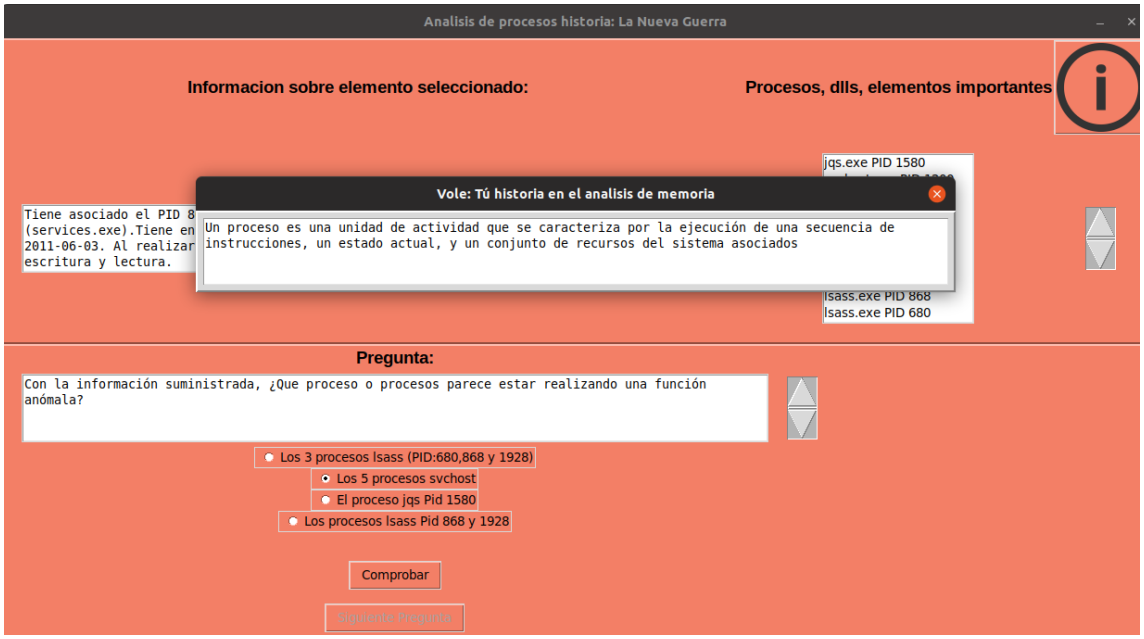

Figura 4.4: Figura ventana info

Tkinter permite asociar a una ventana, una ventana padre, de forma que la ventana creada, bloquea a la ventana padre, permitiendo que en caso de cerrarse la ventana creada vuelva el foto a la ventana anterior. Siendo esta funcionalidad utilizada para la creación de la ventana de información tal y como se muestra en el siguiente código.

```
self.ventanaInfo.protocol("WM_DELETE_WINDOW", self.on_closing)
self.ventanaInfo.transient(master=self.ventanaProcesos)
self.ventanaInfo.grab_set()
self.ventanaProcesos.wait_window(self.ventanaInfo)
```

El desarrollo anteriormente mostrado y detallado cumple el requisito funcional *RF1.3*.

A fin de cumplir el requisito funcional *RF1.4*, se procede a realizar la validación de las preguntas. Para ello, dentro de la ventana de análisis se establecen los controles para que en caso de que el alumno seleccione una respuesta se habilite el botón de comprobar. Este control se establece gracias a la funcionalidad mostrada en el siguiente código.

```
self.radiobutton1 = tk.Radiobutton(marco2, text="Option 1"
, variable=self.valorRespuesta, value=1,
   bg="#2fb45c", command=self.obtener_respuesta_seleccionada)
```

```
    self.radiobutton2 = Radiobutton(marco2, text="Option 2"
        , variable=self.valorRespuesta, value=2,
        bg="#2fb45c", command=self.obtener_respuesta_seleccionada)
    self.radiobutton3 = Radiobutton(marco2, text="Option 3",
        variable=self.valorRespuesta, value=3,
        bg="#2fb45c", command=self.obtener_respuesta_seleccionada)
    self.radiobutton4 = Radiobutton(marco2, text="Option 4",
        variable=self.valorRespuesta, value=4,
        bg="#2fb45c", command=self.obtener_respuesta_seleccionada)

def obtener_respuesta_seleccionada(self):
        self.respuestaSeleccionada = str(self.valorRespuesta.get())
        self.botonValidarRespuesta.config(state="normal")
```

Una vez que el alumno hace *click* en el botón *comprobar* se obtiene el *id* del *radiobutton* seleccionado, y se comprueba si es igual a valor de la clave *respuestacorrecta* del *JSON* de estructura de preguntas para la pregunta actual. Mostrándole al usuario un mensaje informativo indicando si se ha acertado o no la pregunta.

Adicionalmente a este proceso se comprueba si existen más preguntas sin responder, en caso de que existan se habilitara en la interfaz gráfica el botón *SiguientePregunta* el cual se encargara de mostrar la información de la nueva pregunta y las posibles respuesta. En caso de que no existan más preguntas, se modificara la descripción del botón indicando si se desea ir a la siguiente fase.
Para ello es necesario gestionar la información de la fase en la que nos encontramos, así cómo de cual seria la fase siguiente y el numero de preguntas para cada fase, tal y como se indica en el pseudo-código siguiente.

```
def cambiar_pregunta(self):
    Aumento en uno de pregunta siguiente
    Si el tamaño del array de preguntas es superior a la pregunta
    siguiente se muestra la nueva pregunta deshabilitando los
    botones.
  En caso contrario, se comprueba la fase actual
  para eliminar la ventana de esa fase
  y se crea la ventana de la fase siguiente.
```

Quedando la ventana siguiente tal y como se muestra en la figura 4.5

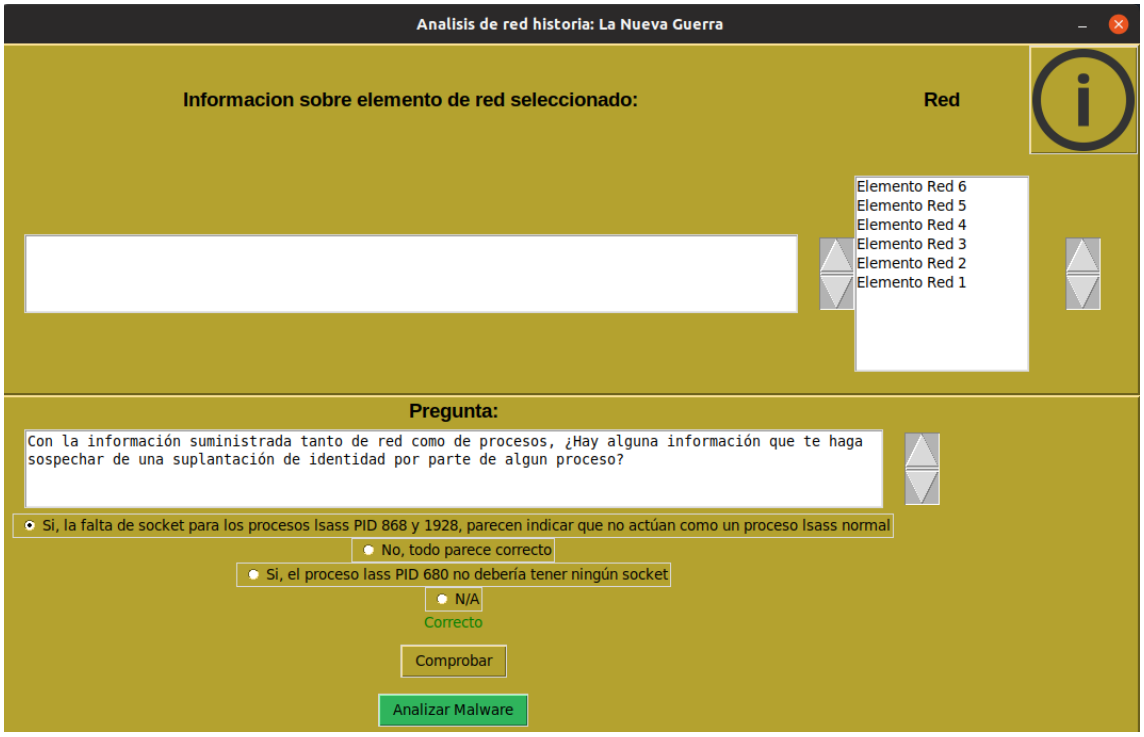

Figura 4.5: Ventana siguiente pregunta

La funcionalidad anteriormente descrita nos permite cumplir el requisito *RF1.4* y *RF1.5*.

4.1.3. Conclusión del análisis y desenlace

Por último, se procede a describir la funcionalidad para la fase de *conclusión*. La interfaz gráfica asociada a esta fase contendrá la información de la conclusión y un botón para finalizar la historia.

La información de la conclusión se obtiene del archivo de texto *resumen.txt* que se encuentra en la carpeta *resumen* anteriormente mencionada en este apartado. Este archivo contiene la información del resumen que sera mostrada, dicha información es no estructurada, siendo texto libre.
A fin de cumplir el requisito funcional *RF1.1*, para crear una nueva historia, sera necesario añadir el nombre de esta al archivo *listahistoria.txt*, crear una carpeta con el nombre de la historia y dentro de esta la estructura de carpetas *analisisProcesos*, *analisisRed*, *analsisMalware*, *resumen* con los archivos mencionados en esta sección y el archivo de texto *descripcion.txt*.

4.2. Crea tu propio análisis

El desarrollo del modo *Crea tu propio análisis* ha sido dividido en 4 bloques principales: Aspectos generales del desarrollo de la aplicación, análisis de procesos, análisis de red y extracción de evidencias. Permitiendo de esta forma realizar un desarrollo encapsulado.

4.2.1. Aspectos generales del desarrollo

En este apartado se abordaran los aspectos generales del desarrollo, como puede ser el diseño general de la aplicación. A fin de cumplir con los requisitos funcionales diseñados en la sección 3.3.5, se han desarrollado 5 interfaces gráficas distintas.

En la primera de ellas, el alumno seleccionara el archivo de memoria sobre el que se va a realizar el análisis de memoria volátil. Inicialmente esta interfaz gráfica contara con 4 botones desactivados, los cuales están diseñados para ir hacia los distintos puntos del análisis de memoria que han sido especificados en el diseño de este modo de uso (análisis de procesos, análisis de red, extracción y reporte). Adicionalmente contara con un botón el cual al pulsarlo muestra una ventana emergente que permite al usuario seleccionar el archivo de memoria.

Al igual que en el modo de uso anterior, las interfaces gráficas han sido desarrolladas con *Tkinter*. En este caso en concreto, *Tkinter* provee de varias funcionalidades para la creación de manera instantánea de interfaces gráficas diseñadas para la selección de archivos, el guardado de archivos o la selección de directorios.
Para la selección de archivos se ha utilizado la siguiente función.

```
askopenfilename(parent=self.ventanaModoExperto)
```

Una vez el alumno ha seleccionado un archivo de memoria, se procede a comprobar si es posible obtener el perfil del sistema operativo asociado a dicha memoria, para ello la interfaz gráfica una vez obtenido el archivo, procede a realizar una llamada a la funcionalidad implementada *obtener_perfil* pasándole la ruta del archivo de memoria. Este método realiza una llamada a la api de *Volatility* al método *imageinfo*.

Para la realización de todas las llamadas a la api de *Volatility* se ha utilizado el método de *Python 3 subprocess.run* tal y como muestra el siguiente código.

```
res = subprocess.run(['vol.py', '-f', filepath,
'imageinfo', '--output', 'json'], capture_output=True)
res = json.loads(res.stdout.decode('utf-8'))
```

La utilización del parámetro *JSON* permite capturar la salida de la llamada al método y serializarlo en un objeto JSON facilitando de esta forma el manejo de la salida como una estructura.
Una vez obtenida la respuesta de la api de *Volatility*, el método *obtener_perfil* se encarga de recorrer el objeto en busca del perfil recomendado. En caso de no existir el método devolverá un perfil vació.
Esta información es transmitida al controlador de la interfaz el cual se encargara, en caso de haber obtenido un perfil, de habilitar los botones relacionados con el análisis,y en caso de no haber podido obtener el perfil mostrara un error al usuario indicándolo que no ha sido posible obtener el perfil del archivo.
La implementación anteriormente descrita permite cumplir el requisito funcional *RF2.1 - Obtener perfil de memoria.*

Para el caso del análisis de procesos y de red se han realizado dos interfaces gráficas iguales donde se encuentra en cada una de ellas una caja de selección donde aparece la lista de funcionalidades que han sido implementadas. Esta caja de selección se nutre de la información almacenada en una variable de *Python* la cual contiene un array con la lista de funcionalidades para cada tipo.

Junto a la caja de selección se encuentra un botón de *ejecutar* el cual se encarga de obtener el valor seleccionado de la caja de selección y comprobar a que método implementado en *vole_api_expansion.py* corresponde, iniciando la llamada.

Se ha implementado un control, para que solo en caso de que la caja de selección tenga un valor seleccionado, se habilite el botón de *ejecutar*. Este control ha sido desarrollado de igual forma que el método explicado para el *Modo historia*, utilizando la funcionalidad de *Tkinter*

```
self.combobox.bind("<<ComboboxSelected>>",self.selection_changed)}
```

Para los casos de análisis de procesos y red, tal y como se ha diseñado en el apartado 3.3, la respuesta de la llamada a la api viene como un array de dos posiciones, la primera contiene la respuesta reducida la cual contendrá la información principal que ayudara a primera vista al alumno, mientras que en la segunda posición del array viene la información completa del método ejecutado.

A fin de poder mostrar estas dos respuesta de manera clara y aportándole versatilidad a la herramienta, la interfaz gráfica para estas dos ventanas cuenta con *notebook* dentro del cual aparecerán dos pestañas, una por cada tipo de salida. De esta forma el cambio de visión entre las respuestas es más ágil, reduciéndose la información que se muestra a cada instante al usuario.

Dando como resultado la siguiente imagen.

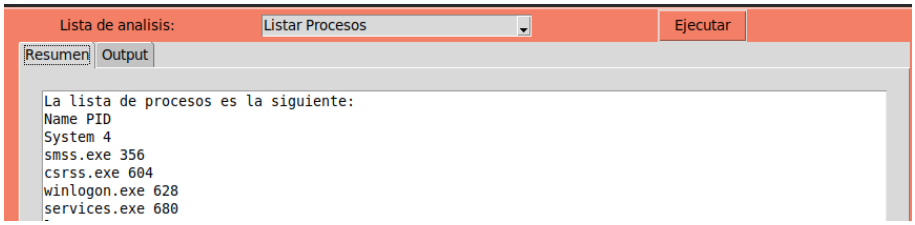

Figura 4.6: Ventana análisis *notebook*

La tercera interfaz gráfica desarrollada corresponde a la interfaz gráfica de extracción de evidencias, la cual también cuenta con una caja de selección de funcionalidades. Una vez seleccionada un tipo de extracción, se comprueba si dicho tipo de extracción requiere la inserción de parámetros adicionales, para lo cual la interfaz gráfica cuenta con una caja de entrada de texto donde el usuario insertara el valor del parámetro.

Pero a diferencia de las interfaces gráficas de análisis de procesos y de red, el botón de *ejecutar* de la interfaz gráfica de extracción de evidencias lanza una ventana emergente para que el usuario seleccione el directorio donde se desea extraer las evidencias. Para ello *TKinter* provee de la siguiente funcionalidad.

```
filedialog.askdirectory(parent=self.ventanaExtraer)
```

Comprobándose inicialmente si en caso de requerir parámetro adicional este fue informado, en caso de no haber sido informado en lugar de la ventana emergente de selección de directorio se mostrara al usuario un mensaje de error.

Una vez seleccionado el directorio, se ejecuta la funcionalidad implementada, mostrándose la información de la extracción en una caja de texto dentro de la ventana de extracción y creándose las evidencias extraídas en el directorio seleccionado.

Por último, se ha diseñado una interfaz gráfica para el proceso de reporte. Dicha interfaz gráfica cuenta con una caja de texto donde se muestra la información del reporte y con un botón de guardado.
Esto botón de guardado lanza una ventana emergente gracias a la funcionalidad *asksaveasfile* de *Tkinter* la cual permite seleccionar el directorio y nombre donde se va proceder al guardado del reporte.

```
def guardar_reporte(self):
    f = filedialog.asksaveasfile(parent=self.ventanaReporte,
    mode='w', defaultextension=".txt")
    if f is not None:
        f.write(self.textoReporte)
        f.close()
```

Permitiéndose cumplir el requisito funcional *RF2.6 - Guardar Reporte* diseñado en la sección 3.3.5 de este documento tal y como se muestra en la imagen siguiente.

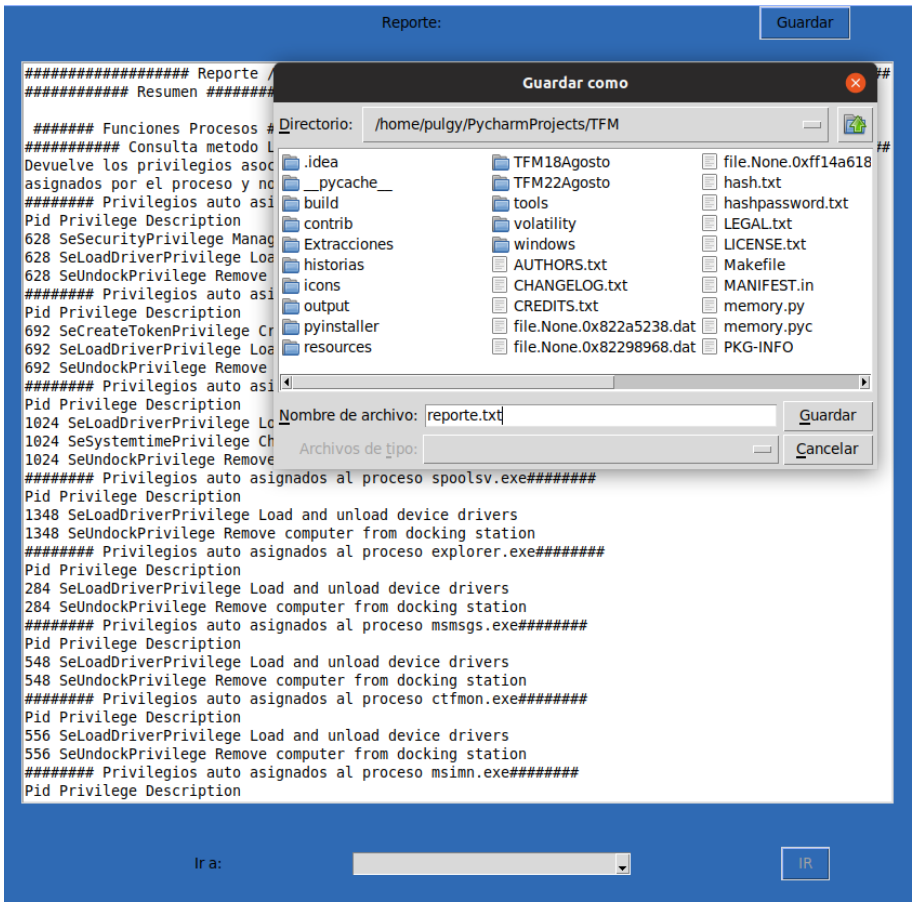

Figura 4.7: Ventana guardar reporte

Cada vez que se ejecuta una funcionalidad de red o proceso se almacena de manera separa la información de la respuesta resumida y de la respuesta completa. Almacenándose en 4 variables distintas esta información dependiendo del tipo de análisis y respuesta (respuesta corta procesos, respuesta completa procesos, respuesta corta red, respuesta completa red).

El reporte se forma de la concatenación de la siguiente información: nombre del archivo de memoria más el conjunto de respuestas cortas del análisis de procesos, informando para cada una de ellas la hora y funcionalidad ejecutada. A todo esto se le suma de igual forma el conjunto de respuestas cortas del análisis de red. Para por último, añadir en otro apartado el conjunto de respuestas completa de procesos y el conjunto de respuestas completa de red.

Quedando de esta forma la información estructurada, mostrando en un primer lugar la información de respuesta más relevante de los proceso ejecutados englobados en procesos o red para posteriormente dar paso al total de la información para un análisis más profundo y detallado. Cumpliéndose por tanto el requisito funcional *RF2.5 - Generar Reporte.*

Estas 4 interfaces gráficas diseñadas cuentan en su parte inferior con una caja de selección que permite al alumno seleccionar hacia que ventana de análisis quiere ir y un botón que le permite cambiar de ventana, facilitando de esta forma la navegación entre ventanas, permitiendo al alumno una mayor versatilidad en el uso de la herramienta.

Tal y como se ha comentado en el capitulo 3 de este documento, el análisis de memoria volátil no es un proceso lineal, por lo que además de la implementación de la funcionalidad que permite navegar entre las interfaces gráficas de análisis, se ha implementado un guardado automático de la ultima funcionalidad ejecutada en cada ventana, de esta forma en caso de volver a una ventana anterior, se mostrara al alumno la información de respuesta del método que fue ejecutado en dicha ventana y el método que fue seleccionado. Para ello se almacenara de manera separada por cada interfaz, el último método seleccionado y el array de respuesta que este devolvió.

Esta funcionalidad de auto guardado solo está implementada en las interfaces de análisis de procesos y análisis de red.

A continuación se procede a explicar la implementación de las funcionalidades respecto al análisis de memoria volátil extraídas del diseño de la herramienta.

4.2.2. Análisis de procesos

Tal y como se describe en la sección 3.3.2 de este documento, el análisis de procesos resulta el eje principal sobre transcurre el análisis de memoria volátil.
Para ello y a fin de cumplir con los requisitos especificados en el diseño, se ha provisto a la herramienta de las siguientes funcionalidades:

- *Listar Procesos*: Para la realización de esta funcionalidad se realiza una llamada a la herramienta *Volatility* en concreto al método *pslist* el cual devuelve el conjunto de procesos.

  Formándose dos respuestas, la reducida incluirá únicamente el nombre e identificador del proceso, mientras que la completa contiene la dirección de memoria, el nombre del proceso, el identificador del proceso, el identificar del proceso padre, el numero de hebras, el numero de manejadores, el identificador de sesión, si el proceso usa 32 bit de espacio de direcciones o 54 y la hora de inicio y fin del proceso.

- *Listar Procesos Ocultos*: Para la realización de esta funcionalidad se han realizado dos llamadas a la herramienta *Volatility*, concretamente a los métodos *psxview* y *pslist*. El método *psxview* nos permite realizar una búsqueda de procesos desde 7 perspectivas distintas, considerándose si un proceso es oculto si no ha sido encontrado en la perspectiva que usa el proceso *pslist* pero si alguno de las otras perspectivas.

  Adicionalmente se realiza una llamada al método *pslist* almacenando solo los procesos que cumplían la condición anterior. Sobre estos procesos se comprueba si alguno de estos procesos tiene más de una hebra o el identificador válido pero en cambio tiene informado el tiempo de salida. Esto indicara que este proceso está intentando camuflarse.

  La respuesta reducida incluirá el nombre e identificador de los procesos ocultos y de los procesos que han intentado camuflarse modificando su tiempo de salida.

  La respuesta completa contendrá la dirección de memoria, nombre del proceso, el identificador del proceso, el tiempo de salida y si se ha encontrado para cada una de las 7 perspectivas.

- *Listar Procesos Críticos*:

  La obtención de procesos críticos se realiza mediante un filtrado de la respuesta del método *pslist*, cogiendo únicamente los procesos cuyo nombre sea Idle, System, csrss.exe, services.exe, svchost.exe, lsass.exe, winlogon.exe o explorer.exe.

- *Listar DLLs Asociadas a Procesos*: La información de la lista de DLLs asociadas a procesos se obtiene mediante la llamada al método *ldrmodules*. A fin de facilitar la búsqueda para el usuario la información de respuesta se encuentra agrupada por procesos. Para ello se obtiene la información devuelta por el método y se comprueba el PID al que está asociado cada DLL, ordenando la salida por PID.

  De esta forma, la información mostrada por la respuesta resumida, permite al usuario de un vistazo comprobar que DLLs están asociadas a cada proceso.

- *Listar DLLs Ocultas*: Se considera que una DLL se encuentra oculta si solo se puede encontrar en memoria, inicializada o cargada. Esta información se puede obtener gracias, al método *ldrmodules*, de tal forma que para implementar esta funcionalidad se realizara un filtrado de los resultados, obteniendo únicamente los que cumplan la condición anteriormente expuesta.

  De igual manera que ocurre en la funcionalidad anterior, la información de respuesta resumida se ha agrupado por proceso.

- *Listar Privilegios auto-establecidos*: El método *privs* de *Volatility* permite obtener todos los privilegios asignados a cada proceso, indicando si el privilegio forma parte de la herencia del proceso padre o si ha sido auto-establecido.

  Para implementar esta funcionalidad se realizara una consulta para obtener todos los procesos almacenados en memoria ejecutando para cada uno de ellos el método *privs* y filtrando únicamente los privilegios que tengan el valor *Present,Enabled*, lo que indicaría que han sido auto-establecidos.

  La respuesta reducida de esta funcionalidad agrupara la información por procesos, indicando si el privilegio auto asignado y la descripción del mismo.

- *Listar Drivers y tabla IRP*: La obtención de los drivers cargados y las tablas IRP se realizan mediante el escaneo del espacio de memoria buscando la etiqueta *DRIVER_OBJECT*. *Volatility* provee de esta funcionalidad mediante el método *driverirp*.

- *Listar Registros Persistentes*: Para la búsqueda de claves de registro persistente, se realizara la obtención de las claves de registro:

  Microsoft\Windows\CurrentVersion\RunOncen
  Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
  Microsoft\Windows\CurrentVersion\Run
  Microsoft\Windows NT\CurrentVersion\Windows
  Microsoft\Windows NT\CurrentVersion\Windows\Run
  Microsoft\Windows\CurrentVersion\Run
  Microsoft\Windows\CurrentVersion\RunOnce

  El método *printkey* permite obtener el valor de una clave de registro informando el nombre de la esta. Posteriormente con la obtención de dicho valor se comprueba si para alguna de ellas existe algún proceso, ejecutable o DLL asociado.

- *Listar Persistencia por servicios*: Para la obtención de la lista de servicios que intentan establecer persistencia es necesario obtener el registro asociado a *currentcontrolset* obteniendo el valor asociado a la clave *REG_INK* que contenga dentro de su valor la cadena de texto
  ControlSet. Una vez detectado este valor se procede a extraer de este campo el valor del *controlset*.

  En caso de poder obtener este valor se procede a buscar el contenido de las claves pertenecientes a ese *controlset* concatenado con el valor

services. Permitiendose obtener el nombre de los servicios que se encuentran en el *controlset*.

Por cada uno de estos servicios obtenidos se vuelve a realizar una búsqueda de las claves de registro asociados a la clave *services nombredelservicio*, buscándose sobre los resultados obtenidos las claves con nombre *REG_SZ* que contengan un valor distinto de - y mostrando ese conjunto de servicios encontrados al usuario.

- *Historial de comandos y consola*: La obtención del historia de comandos y consola ejecutados, se realizara mediante la llamada al método *consoles* de la herramienta *Volatility*.

  A fin de facilitar el análisis al usuario, la respuesta resumida mostrara únicamente los comandos ejecutados sin informar la respuesta que estos producen.

4.2.3. Análisis de red

Tal y como se describe en la sección 3.3.3 de este documento, el análisis de red permite al alumno conocer el conjunto de conexiones que se han establecido en la maquina objeto del análisis, así como las distintas paginas que han sido visitadas. Para ello y a fin de cumplir con los requisitos especificados en el diseño, se ha provisto a la herramienta de las siguientes funcionalidades:

- *Conexiones Remotas*: La obtención de las conexiones remotas activas en el análisis de memoria volátil, se realiza mediante la llamada al método *connections*, mostrando al usuario la información relativa al proceso que establece la conexión, dirección local y dirección remota para la respuesta reducida.

- *Tarjetas de Red en modo Promiscuo*: Para la obtención y detección de tarjetas de red en modo promiscuo se realizara una búsqueda de todos los socket abierto gracias al método *sockets* de los cuales se filtrara por los que utilicen el protocolo 0 *HOPOPT* y que estén asignados al puerto 0, lo cual indicara que está en modo escucha.

- *Conexiones Remotas Antiguas*: La obtención de las conexiones remotas antiguas que ya no se encuentran activas, se realiza mediante la llamada a dos métodos *connscan* y *connections*. *connscan* devuelve el conjunto de conexiones remotas activas e inactivas, por lo que se realiza un filtrado eliminando las conexiones encontradas en la llamada a *connections*.

  La información obtenida puede estar parcialmente corrompida ya que ha podido ser sobreescrita por el sistema.

- *Urls en procesos de navegadores*: Para la obtención de urls que han sido consultadas desde los navegadores es necesario listar el conjunto de procesos que se encuentran en el sistema, obteniendo si existen, los identificadores para los procesos con nombre iexplore.exe, firefox.exe, chrome.exe, Chrome.exe, Firefox.exe y MicrosoftEdge.exe, ya que son los nombres de los procesos asociados a navegadores más utilizados actualmente en Windows.

  Por cada uno de los identificadores obtenidos se realiza un escaneo por fuerza bruta en el espacio de memoria utilizando el método *yarascan* de *Volatility* el cual informando un identificador de proceso permite realizar búsquedas utilizando estructuras.

  En esta implementación se ha utilizado la siguiente regla para la búsqueda de estructuras similares a URLs.

```
(https?:\/\/(?:www\.|(?!www))[a-zA-Z0-9][a-zA-Z0-9-]+[a-zA-Z0-9]\.
[^\s]{2,}
|www\.[a-zA-Z0-9][a-zA-Z0-9-]+[a-zA-Z0-9]\.[^\s]{2,}|https?:\
/\/(?:www\.|
(?!www))[a-zA-Z0-9]+\.[^\s]{2,}|www\.[a-zA-Z0-9]+\.[^\s]{2,})/
```

- *Historial Internet Explorer*: Adicionalmente a la funcionalidad anterior, *Volatility* tiene implementado un *plugin* gratuito llamado *iehistory*, el cual permite realizar una búsqueda del historial de navegación de Internet Explorer a partir de la recuperar de los ficheros de cache asociados a este proceso.
- *Recuperar DNS Cache*: Para la recuperación del archivo de cache de la DNS es necesario consultar el conjunto de archivos que se encuentran en memoria, para ello se utiliza la llamada al método *filescan*, sobre el conjunto de archivos obtenidos se filtra aquellos que en su dirección contengan la cadena de texto *\host*. Obteniendo la dirección asociada a estos archivos.

  Por cada una de estas direcciones se realiza una extracción del archivo en cuestión utilizando el método *dumpfiles* pasándole como parámetro la dirección de memoria. El problema es que el método *dumpfile* guarda el archivo en una dirección local pero desconocemos el nombre de este archivo. Por lo tanto, al hacer la llamada al método *dumfile* se debe obtener el *sumary* en el cual se obtiene si la información se ha extraído correctamente y el nombre del archivo que se ha creado.

  Una vez obtenido el nombre del archivo, la funcionalidad implementada lee el contenido y lo muestra al usuario.

```
res = subprocess.run(['vol.py', '-f', filepath, 'dumpfiles'
'-Q', direccion, '-D',directorio, "-S", sumary],capture_output=True)
```

4.2.4. Extracción de evidencias

Tal y como se describe en la sección 3.3.4 de este documento, la extracción de evidencias resulta un elemento indispensable en el análisis de memoria volátil, permitiendo obtener evidencias para un posterior análisis individual.
Para ello y a fin de cumplir con los requisitos especificados en el diseño, se ha provisto a la herramienta de las siguientes funcionalidades:

- *Todas las DLLs*: *Volatility* provee de un método llamado *dlldump* el cual permite extraer las DLLs asociadas a una memoria. La implementación de esta funcionalidad consiste en llamar a este método, obteniendo la respuesta del mismo, el cual permite conocer que DLLs se han podido extraer correctamente y cuales no. Es necesario informar el directorio local donde se desea extraer los archivos.

- *DLLs Asociadas a Proceso*: De igual forma que el método anterior, *dlldump* permite pasarle como parámetros un identificador de proceso, extrayendo únicamente las DLLSs asociados a este proceso.

- *DLLs Espacio de Memoria*:. De igual forma que el método anterior, *dlldump* permite pasarle como parámetros la dirección de memoria, extrayendo únicamente la DLL asociada a ese espacio de memoria.

- *Todos los Drivers*: Para la extracción de drivers se ha utilizar el método *moddump* el cual permite extraer el conjunto total de drivers que se encuentran asociados a la la memoria volátil. Esta implementación no modifica la salida del método *moddump*, sirviendo únicamente como interfaz gráfica del mismo.

- *Obtener hash contraseñas*:

  La obtención de los *Hash* de contraseñas del dominio de usuario se realiza mediante la llamada al método *hashdump*, el cual ha sido implementado por *Volatility* para la obtención de los hash que contienen la información de las contraseñas de los usuarios del dominio de la memoria volátil.

  Estas contraseñas se encuentran cifradas por lo que se realiza una llamada a la herramienta *John the ripper* que mediante la fuerza bruta, es capaz en algunos casos de realizar el desencriptado de las contraseñas.

Finalmente se mostrara a los usuarios el *hash* obtenido y el resultado de su intento de desencriptado.

La implementación de este conjunto de funcionalidades permite cumplir los requisitos funcionales *RF2.2 - Analizar procesos*,*RF2.3 - Analizar elementos de red* y *F2.4 - Extraer Evidencias* definidos en sección 3.
El conjunto de todas las funcionalidades necesarias este modo de uso han sido desarrolladas en el archivo*vole_ api_ expansion.py* y *vole.py*.

4.3. Dependencias

La utilización de lenguajes y herramientas externas provoca que la solución propuesta tenga dependencias siendo necesario que se encuentren instaladas en el sistema que alojara la herramienta las siguientes herramientas y bibliotecas:

- *Python* versión 3.7: Sera necesario tener instalado esta versión que sera la encargada de inicializar la herramienta (vole.py).
- *Python* versión 2.7: Sera necesario tener instalado esta versión ya que es necesaria para instalar y ejecutar la herramienta *Volatility*.
- *John the Ripper* versión 1.9: Es necesario instalar esta herramienta para realizar el desencriptado de las contraseñas extraídas por la herramienta desarrollada.
- *Tkinter* versión 8.5: Paquete de *Python* con el que se ha desarrollado la interfaz gráfica de la herramienta.
- *Volatility* versión 2.6: Sera la herramienta encargada de surtir a la herramienta desarrollada del conjunto de funcionalidades primarias para el análisis de memoria volátil. Sera la base sobre la que se construyan las funcionalidades propias de la herramienta desarrollada para el trabajo fin de máster.

  Volatility debe ser instalada bajo *Python* versión 2.7 o inferior ya que actualmente no tiene soporte para las versiones de *Python* 3.0 y superior. *Volatility* tiene dependencia con los paquetes *Distorm3*[16], *Yara*[17], *PyCrypto*[18], *PIL*[19], *OpenPyxl*[20] y *ujson*[21]. Sera necesario por tanto realizar la instalación de estos paquetes para completar todos requisitos necesarios para la utilización de *Volatility*.

La versatilidad que aporta el desarrollo en *Python* permite que la herramienta sea utilizada en los sistemas operativos Windows, Mac OS y Linux.

La herramienta desarrollada en este trabajo fin de máster, *Vol-E* se encuentra disponible en el repositorio `https://github.com/JavierOrdMartin/vol-e`

Capítulo 5

Validación

En este apartado se pretende realizar una validación de las principales funcionalidades desarrolladas en el capitulo 4 de este documento. Dividiéndose la validación en los dos modos de uso que tiene la solución desarrollada.

5.1. Modo historia

Para este apartado se realizara las validaciones relacionadas con el modo historia.

A fin de realizar la validación del modo historia, se ha añadido una historia al conjunto de datos iniciales de la herramienta.

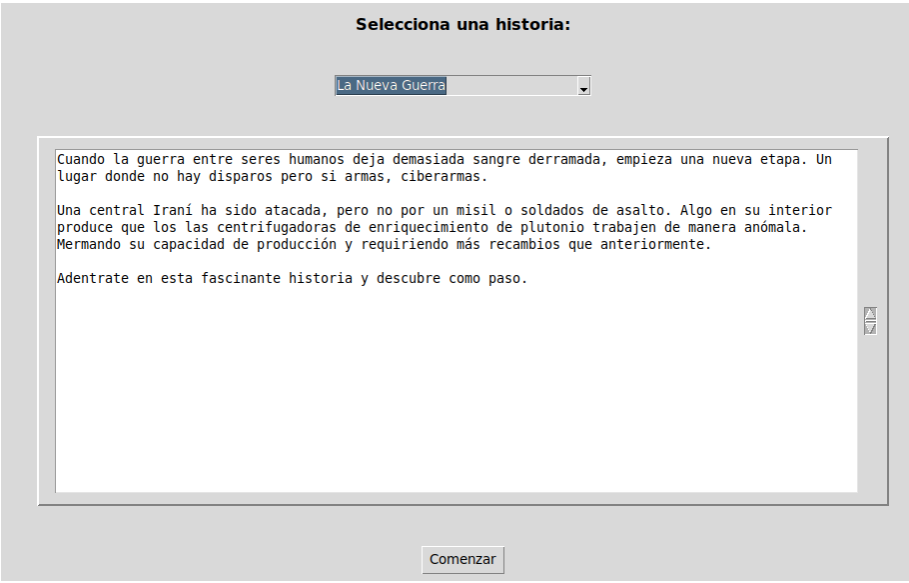

Figura 5.1: Ventana Selección Historia

Para la creación de esta historia, se ha tomado el análisis de memoria volátil sobre el incidente ocurrido en las centrifugadoras de plutonio de la central nuclear de Irán, donde el malware *Stuxnet* altero el correcto funcionamiento de esta central.
Para la obtención de evidencias y conclusiones sobre el incidente ocurrido se ha tomado como referencia el análisis de memoria volátil realizado por *behindthefirewalls* [22], adicionalmente para entender el contexto histórico de este malware se han obtenido referencias de *genbeta*[23].

La validación del modo historia empieza listando el conjunto de historias existentes, como se muestra en la Figura 5.1, solamente existe la historia anteriormente mencionada, donde aparece la descripción añadida al documento de descripción de dicha historia.
Una vez seleccionada la historia y pulsado el botón de *Comenzar*, se muestra la ventana de análisis de procesos, donde aparecen listados el conjunto de elementos extraídos relacionados con el análisis de procesos junto con su descripción y la primera pregunta sobre este análisis.

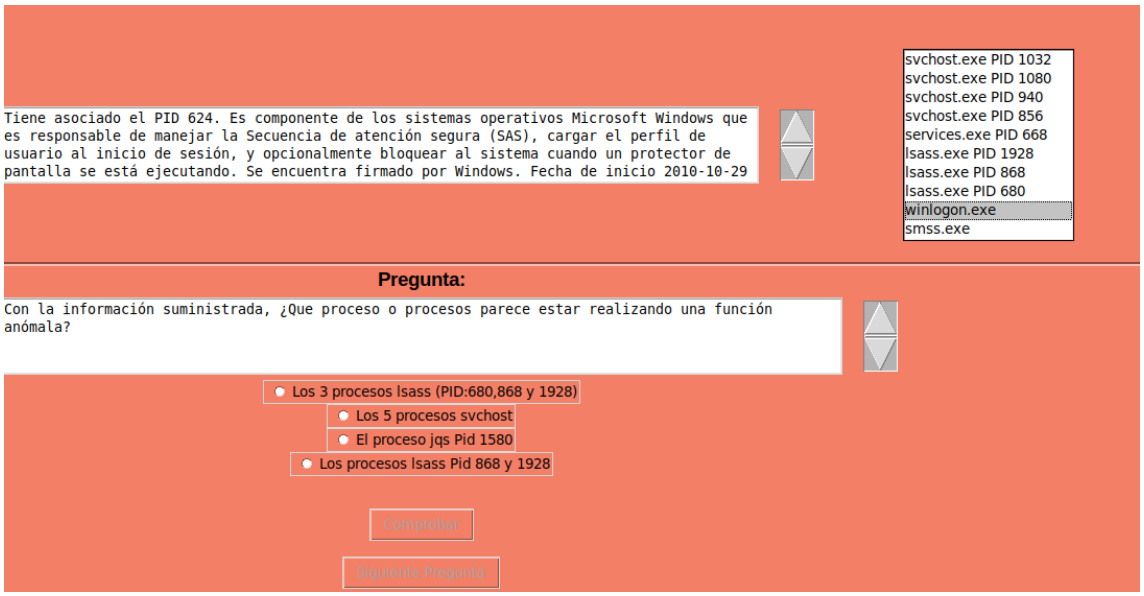

Figura 5.2: Ventana Análisis procesos Stuxnet

Tal y como se muestra en la Figura 5.3, al seleccionar una respuesta y comprobar si es correcta, aparece un mensaje informando si la respuesta seleccionada es la correcta o no. Para esta primera validación la respuesta seleccionada no es la correcta.

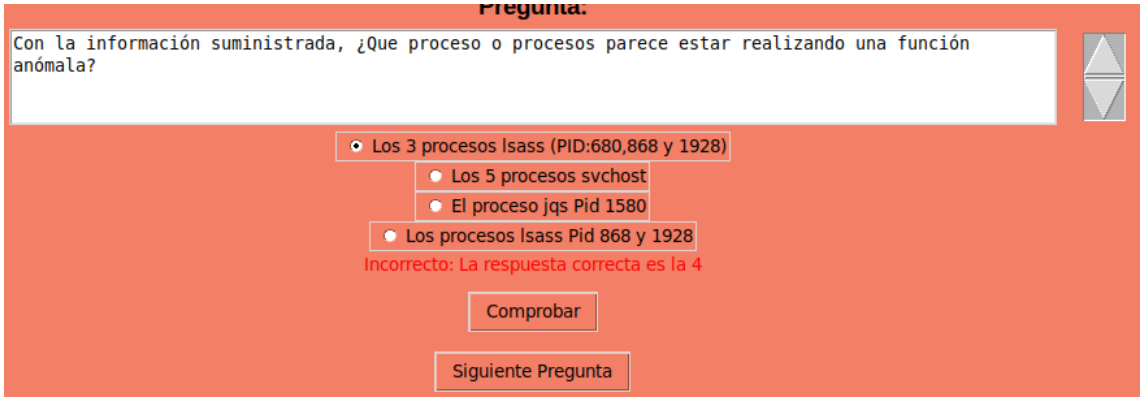

Figura 5.3: Validación pregunta incorrecta

Una vez comprobada una pregunta, la herramienta comprueba que existen ,más preguntas, permitiendo cambiar de pregunta. En la siguiente figura se muestra la siguiente pregunta una vez pulsado el botón *Siguiente Pregunta*, esta nueva pregunta ha sido respondida de manera correcta, por lo que aparece un mensaje informativo indicando que la comprobación ha sido correcta.

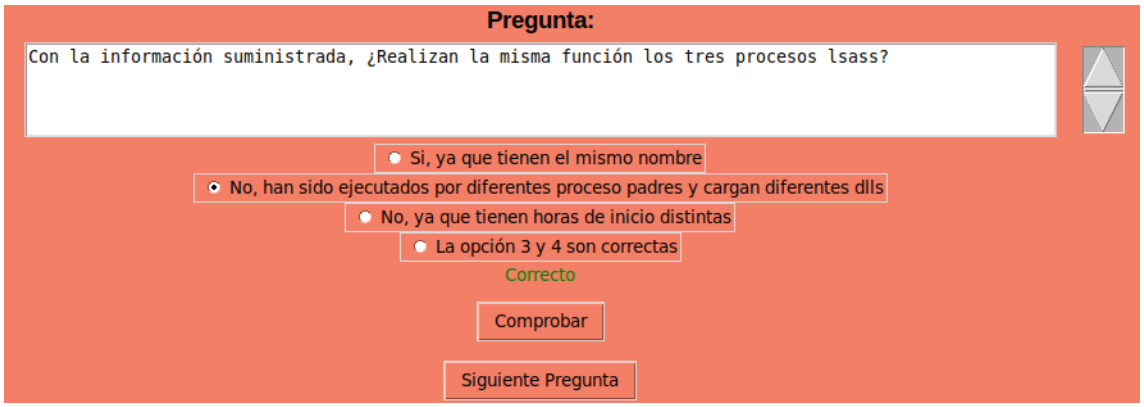

Figura 5.4: Validación pregunta correcta

Una vez respondido el conjunto total de preguntas para una fase de análisis, desaparece el botón *Siguiente Pregunta*, dejando paso a un botón indicando el nombre de la siguiente fase.

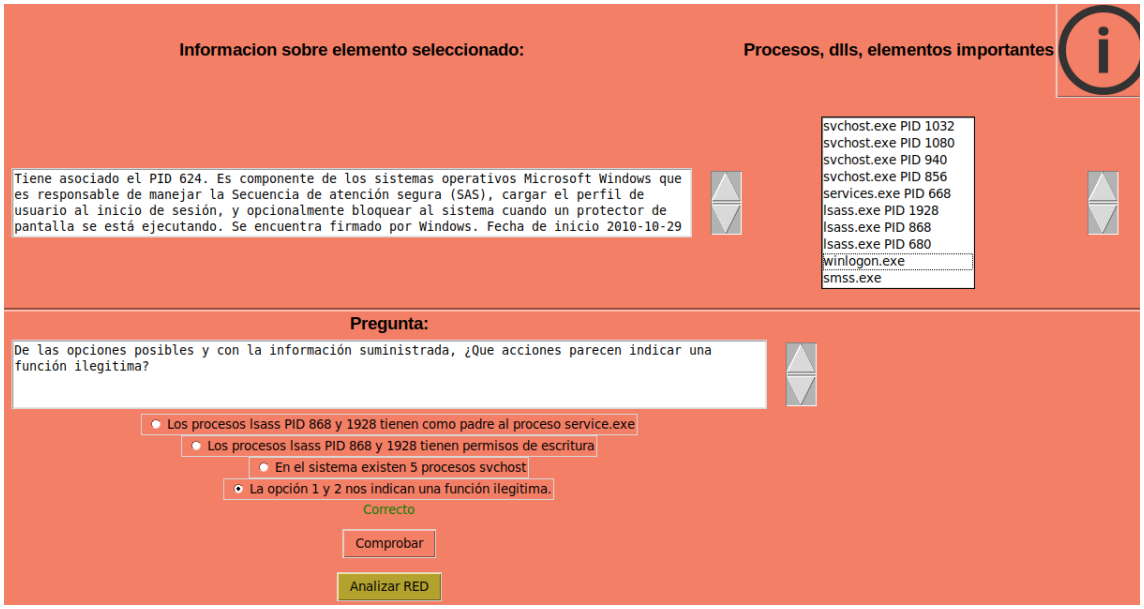

Figura 5.5: Validación cambio fase

Este nuevo botón nos permitirá avanzar en la historia, en este caso en concreto, hacia el análisis de red de la historia. Tal y como se ve en la siguiente imagen aparece la información relacionada con el análisis de red de la historia *stuxnet*.

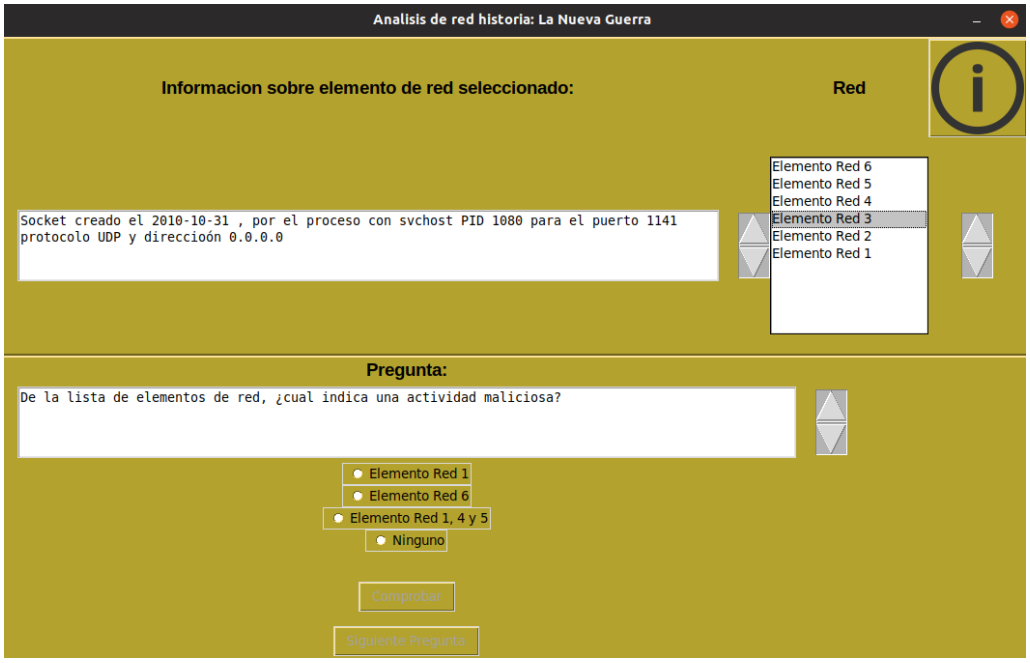

Figura 5.6: Validación cambio fase Red

De igual forma como ocurre en la fase de procesos, cuando se responden al conjunto total de preguntas del análisis de red, permite al alumno avanzar en la historia hacia el análisis de malware.

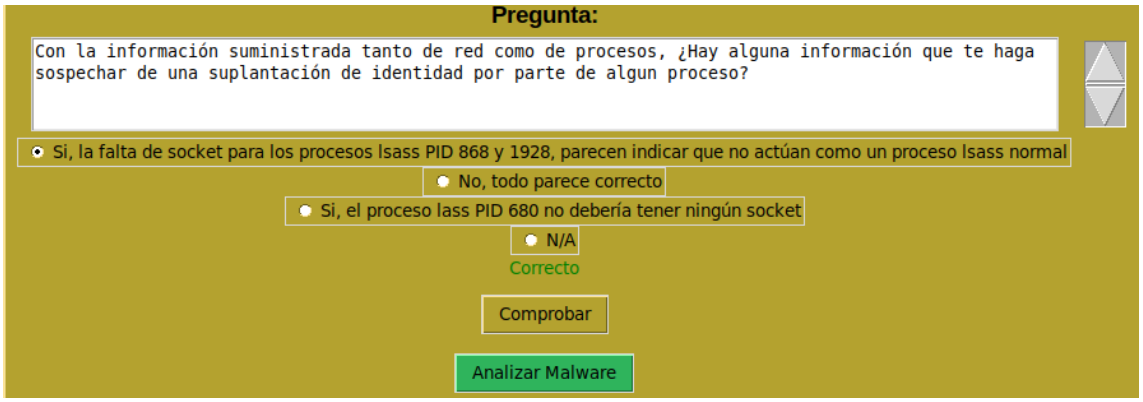

Figura 5.7: Validación cambio fase desde análisis Red

Una vez mostrada la información relativa al análisis de posibles evidencias malware y contestado el total de preguntas que han sido añadidas para esta fase, la herramienta permite al alumno avanzar hacia una conclusión/resumen del incidente.

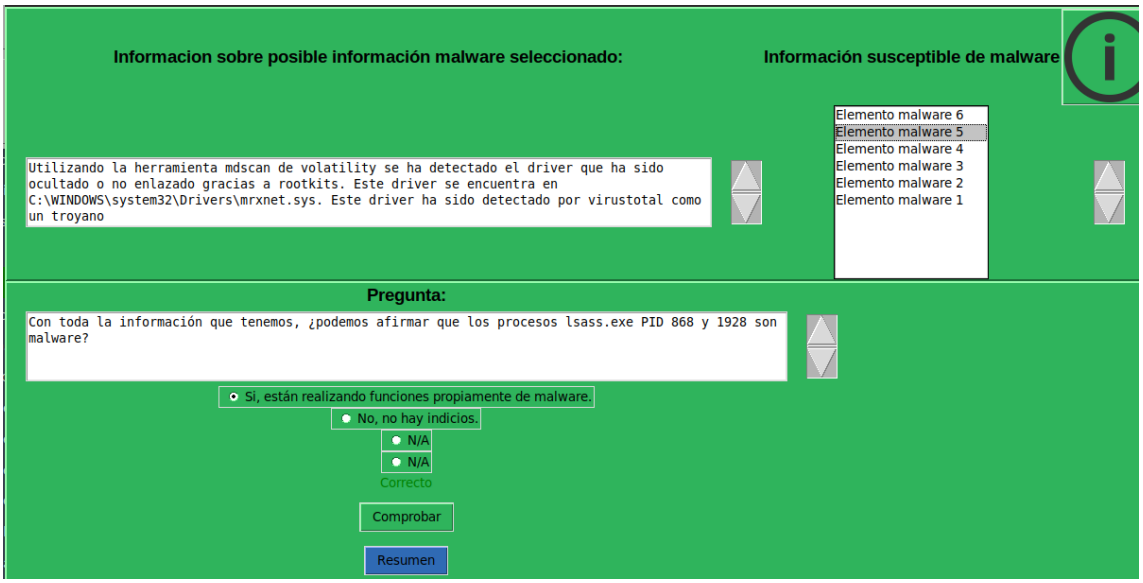

Figura 5.8: Validación cambio fase desde análisis malware

La última fase del análisis perteneciente a la conclusión o resumen desvela al usuario todo lo ocurrido a lo largo de la historia.

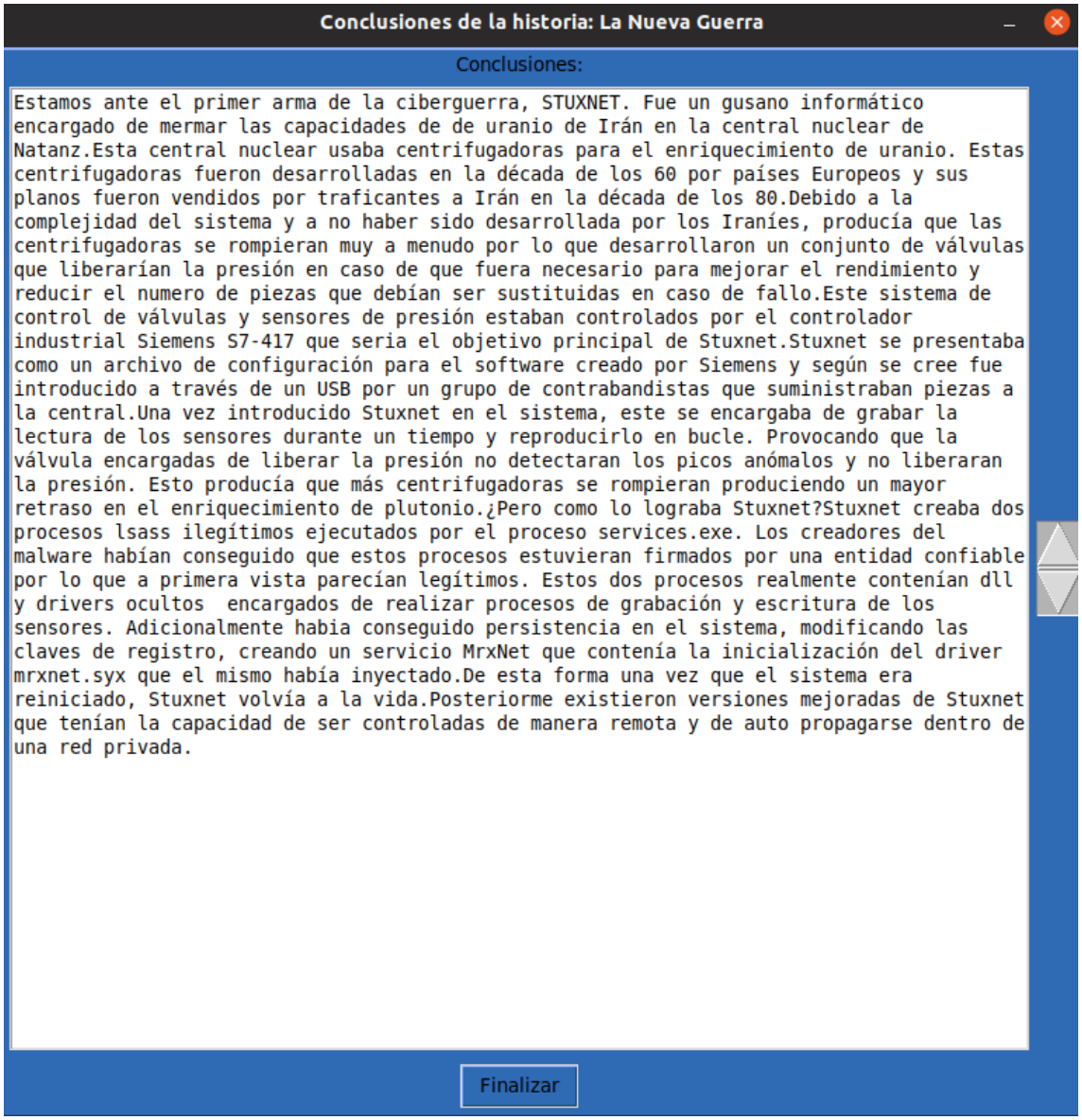

Figura 5.9: Validación ventana resumen

La información relativa a los diverso análisis, preguntas, descripción de la historia historia, se encuentran en los incluidos en la carpeta *La Nueva Guerra*, que se encuentra en la carpeta *Historias* incluidas en el desarrollo de este trabajo fin de máster.

5.2. Crea tu propio análisis

En este apartado, se realizara las validaciones relacionadas con el modo crea tu propio análisis diseñado y desarrollado en los apartados anteriores.

A fin de realizar una validación a alto nivel de la herramienta desarrollada se ha utilizado una memoria de un sistema operativo Windows WinXPSP2x86 en concreto el archivo de memoria *sample001.bin* obtenido de la pagina *memoryanalysis*[24].

El primer paso a validar es el inicio del modo crea tu propio análisis, tal y como se muestra en la imagen siguiente, todas las funcionalidades se encuentran desactivadas, únicamente permitiendo seleccionar el directorio de una archivo de memoria volátil.

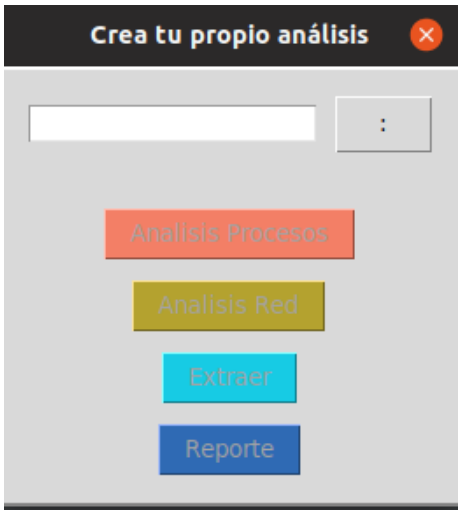

Figura 5.10: Validación ventana crea tu propio análisis

Para el primer caso, se va a seleccionar un archivo de memoria invalido, comprobando tal y como se muestra en la imagen 5.11 no permite realizar ninguna funcionalidad de análisis. Informando al alumno que no se ha podido obtener un perfil valido a partir de un mensaje de error claro y llamativo.
Posteriormente se procede a seleccionar el archivo de memoria *sample001.bin* anteriormente mencionado. Sobre este archivo valido de memoria se ha podido obtener el perfil del sistema operativo, permitiendo la herramienta desarrollada acceder a las funcionalidades de análisis, exportar y reporte. Tal y como se muestra en la Figura 5.12.

Figura 5.11: Validación ventana crea tu propio análisis archivo erróneo

Figura 5.12: Validación ventana crea tu propio análisis archivo correcto

Una vez obtenida una memoria con un perfil valido, se procede a realizar un análisis de procesos, mostrándose el conjunto de funcionalidades implementadas y diseñadas en los apartados anteriores de este documento.

A fin de no extender en exceso la memoria, se procede a ejecutar únicamente dos funcionalidades implementadas en el análisis de procesos.

En primer lugar, se procede a realizar una consulta de los procesos que se encontraban activos en el momento de la extracción de la memoria volátil. Tal y como se muestra en la figura 5.13, la funcionalidad devuelve el conjunto de procesos, en este caso en la salida resumida se muestra la información principal donde se incluye únicamente nombre e identificador del proceso. Pudiendo el alumno encontrar de esta forma la información principal con un vistazo rápido.

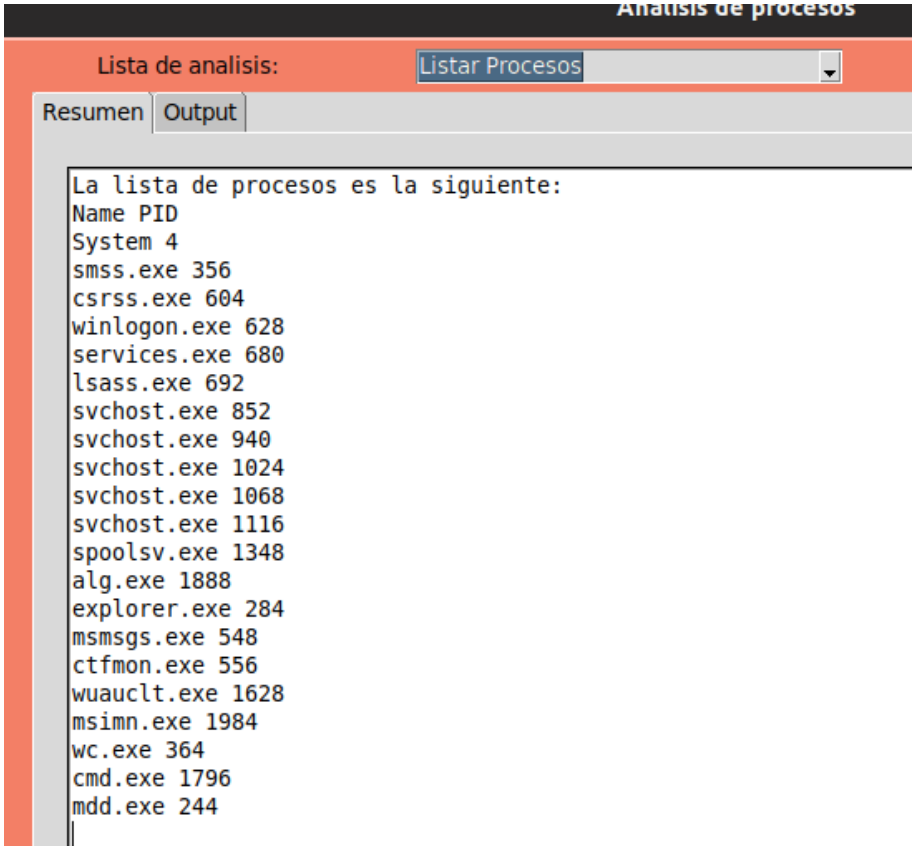

Figura 5.13: Validación ventana análisis procesos, listado de procesos

Adicionalmente se ha realizado una ejecución de la funcionalidad que permite ver los privilegios auto establecidos por los procesos.

En esta primera imagen se muestra la salida reducida, esta incluye la información estructurada por procesos, ayudando al usuario a realizar un análisis de manera rápida y eficaz. Permitiendo comprobar por cada proceso, cuales son los privilegios que han sido auto asignados y la descripción de estos.

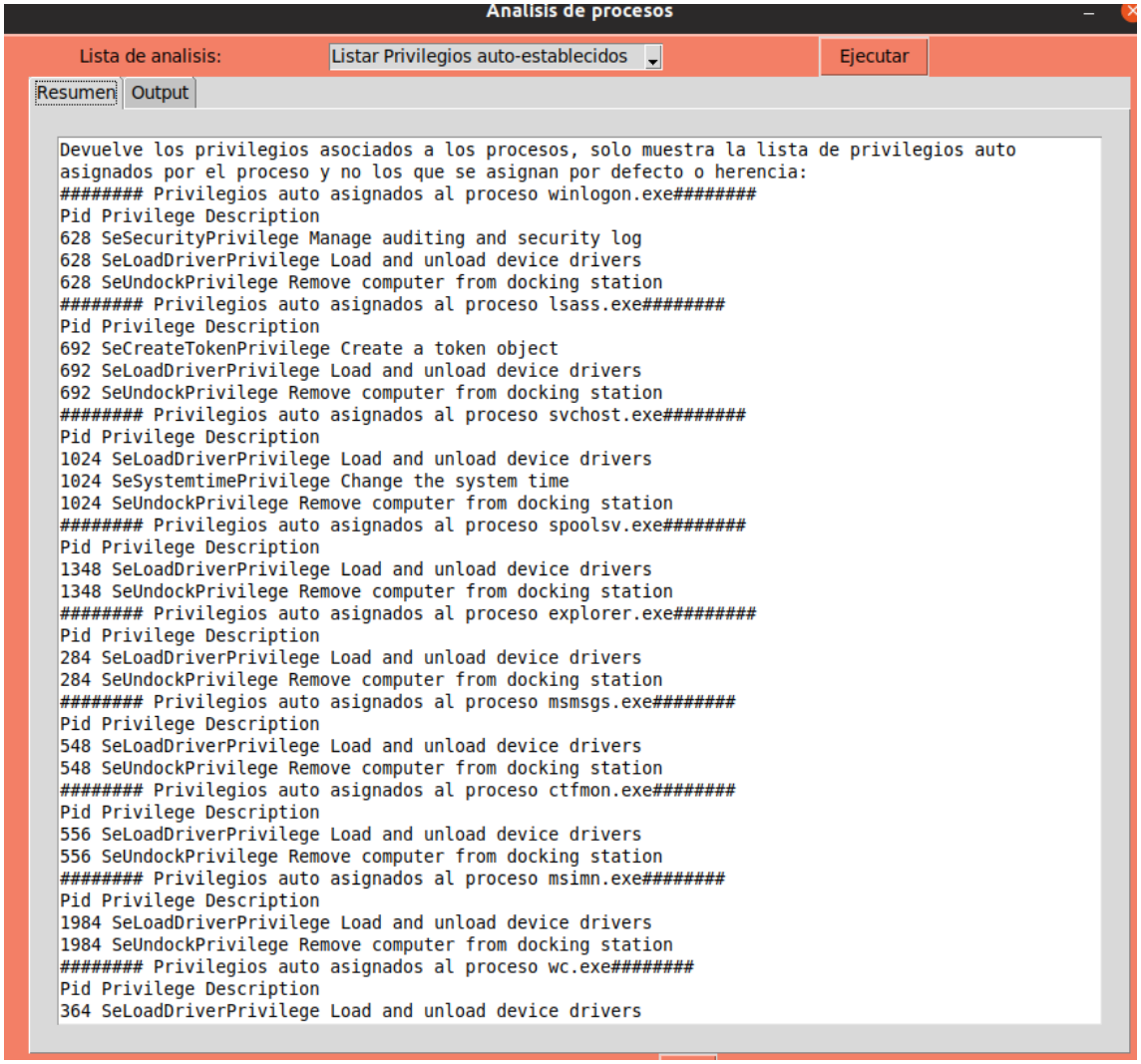

Figura 5.14: Validación ventana análisis procesos, listado privilegios resumen

En contra posición, la salida completa, muestra toda la información que puede servir al alumno para obtener información adicional. El cambio de visualización de la respuesta por parte del alumno se realiza mediante un sencillo click en el modo de respuesta que desea ver, permitiendo una transición rapida entre ambas, tal y como se muestra en la figura 5.15.

También se pueden observar, es la diferenciación del color de la ventana según el tipo de análisis que esta realizando el alumno. Facilitando de esta forma la agrupación de conceptos.

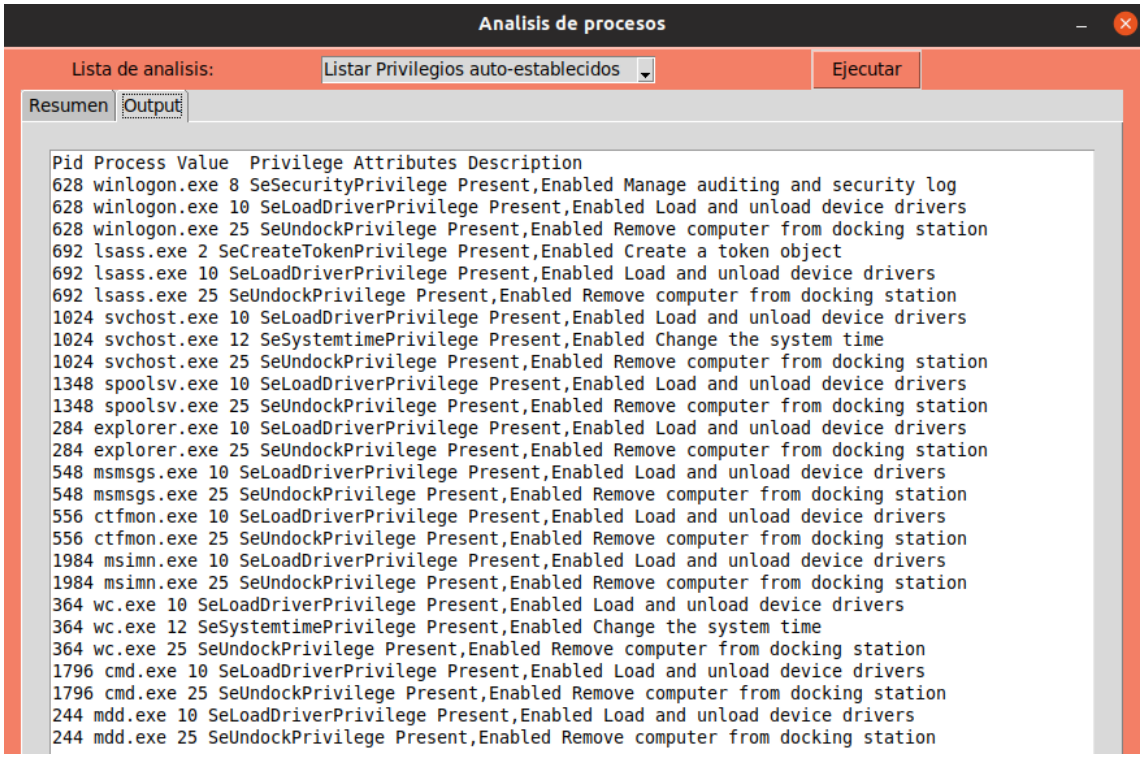

Figura 5.15: Validación ventana análisis procesos, listado de privilegios salida completa

Otro de los aspectos que favorece el análisis de procesos es la navegación entre ventanas, permitiendo desde cualquier tipo de interfaz navegar hacia otro análisis, extracción o reporte. Tal y como se muestra en la Figura 5.16.

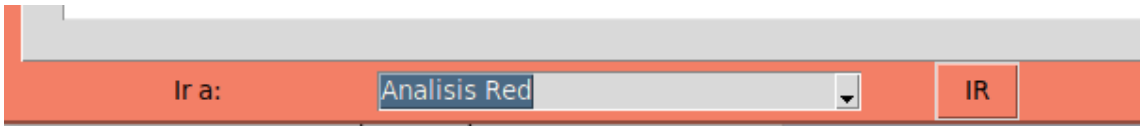

Figura 5.16: Validación ventana análisis procesos navegación

En el caso realizado, se ha navegado desde la ventana de análisis de proceso hacia la ventana de análisis de red, tal. Donde se procede a realizar un análisis de las conexiones que se encontraban activas en la memoria. Un ejemplo se muestra en la Figura 5.17

La solución propuesta, está desarrollada de tal manera que almacena la última información de la funcionalidad de cada tipo de ventana de análisis, permitiendo no perder la información en caso de navegar entre las interfaces. De modo que si volvemos

a la ventana anterior el alumno puede observar la salida de la última funcionalidad ejecutada.

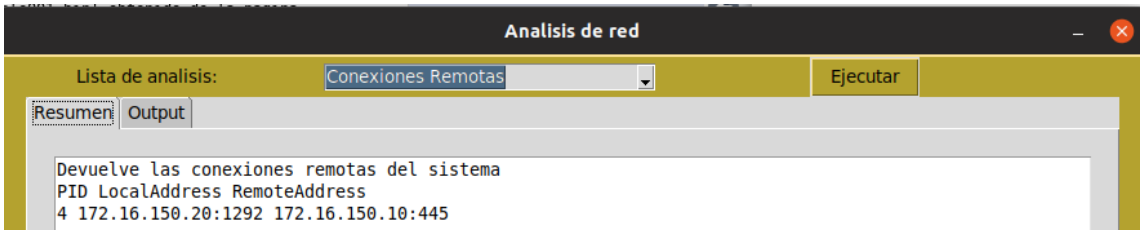

Figura 5.17: Validación ventana análisis red

Otro de los aspectos clave desarrollado, es la extracción de evidencias por parte del usuario, para ello una vez seleccionado la evidencia a extraer y pulsado el botón de ejecutar, se abrirá una ventana emergente donde el usuario deberá indicar el directorio donde se almacenaran las evidencias extraídas.

En la imagen siguiente, se puede observar como se ha seleccionado la extracción de todas las DLLs cargadas y establecido el directorio donde se guardaran.

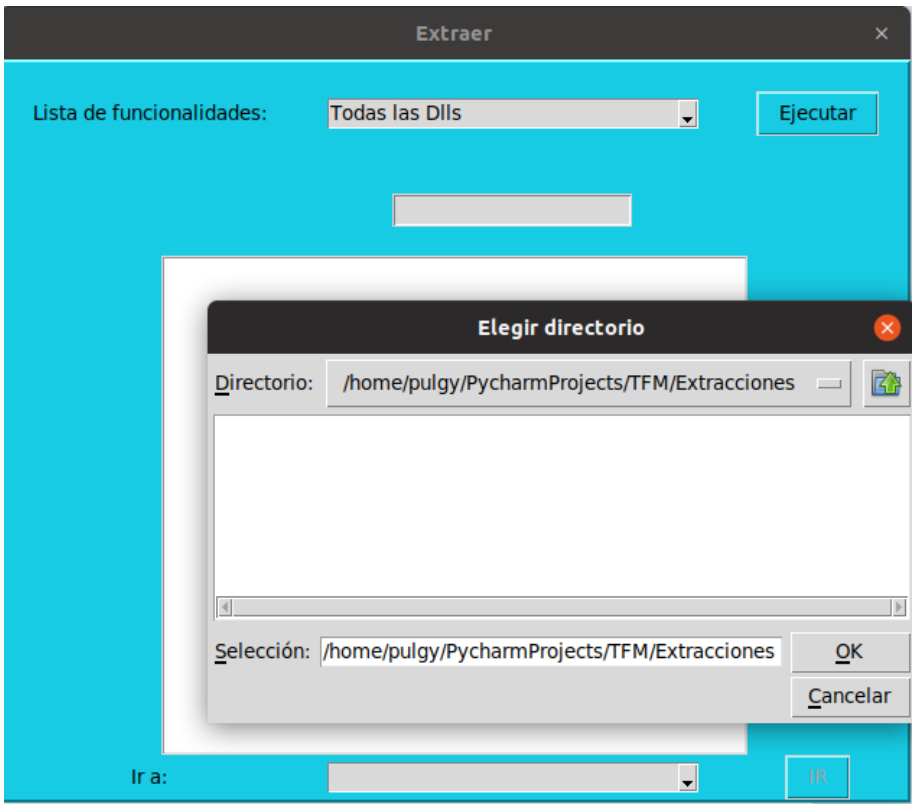

Figura 5.18: Validación ventana extracción selección de directorio

Una vez indicado el directorio y finalizado el proceso, la extracción muestra el resultado del conjunto de DLLs que han sido podido extraídas y cuales no. Al revisar el directorio establecido se comprueba que existen las DLLs indicadas en la salida. Tal y como aparece en la Figura 5.19.

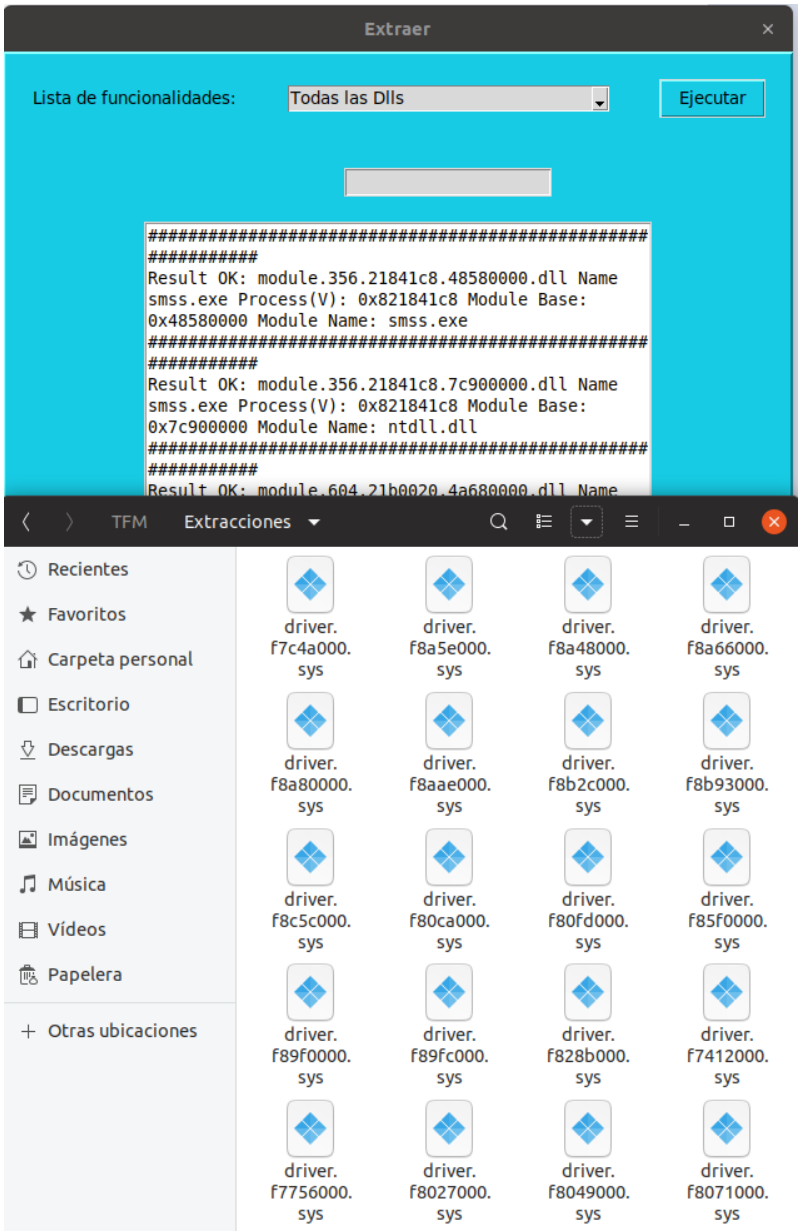

Figura 5.19: Validación ventana extracción DLLs

Las funcionalidades que han sido ejecutadas para el análisis de procesos y red, quedan reflejadas en la ventana de reporte, estructurando las salidas tal y como se muestra en la siguiente imagen.

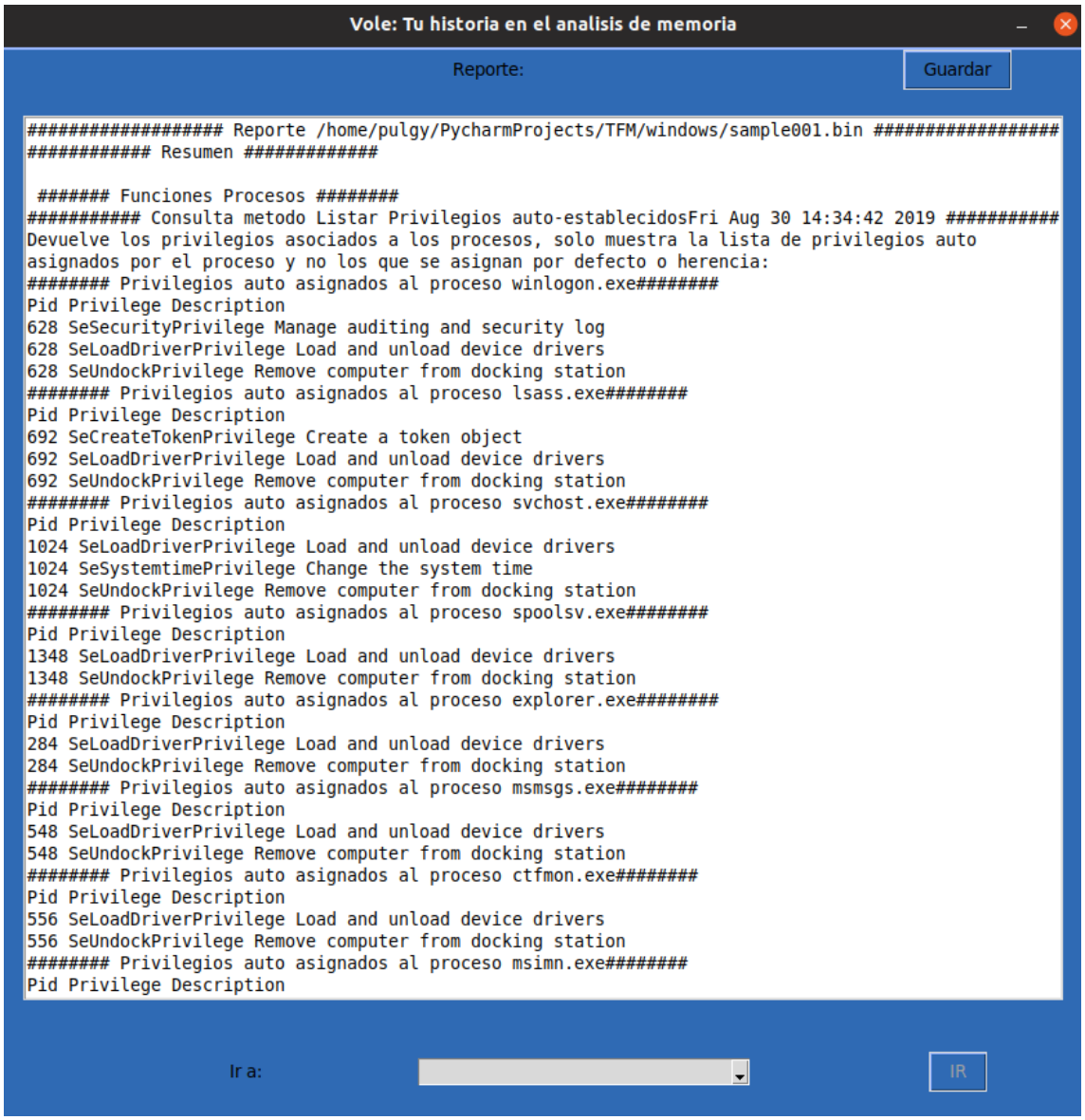

Figura 5.20: Validación ventana reporte

La interfaz gráfica de reporte, permite al alumno guardar el reporte para un posterior análisis en este caso, se ha guardado el reporte bajo el nombre *reporteejemplo.txt* mostrando este la misma información que aparecía en la ventana de reporte de la imagen anterior.

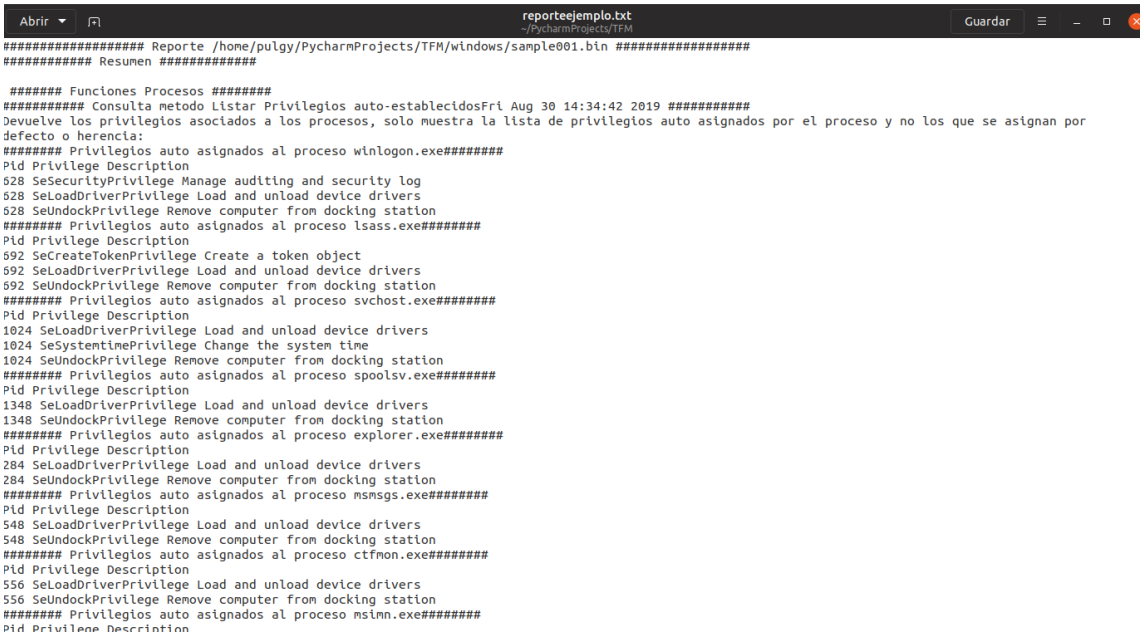

Figura 5.21: Validación reporte guardado

Capítulo 6

Conclusiones

6.1. Objetivos cumplidos

Los objetivos establecidos al comienzo de este TFM han sido completados de manera satisfactoria.

El objetivo 1 se ha cumplido y expuesto en el apartado 2, realizando un análisis de mercado permitiendo conocer las herramientas actuales y las necesidades actuales del mercado con respecto al análisis de memoria volátil. Dentro de este apartado y en conjunto con el apartado 3 se realiza un análisis de las necesidades principales en el análisis de memoria volátil y de as funcionalidades más utilizadas, cumpliéndose por tanto las necesidades del objetivo 2.

La herramienta desarrollada de *Python* cumple con las especificaciones y necesidades establecidas en el objetivo 3. Cumpliéndose por tanto las necesidades de este, gracias a la realización de una herramienta para el apoyo en el ámbito del análisis de memoria volátil, que permita no solo para la realización de análisis si no para la formación de futuros profesionales. Esta herramienta tal y como se describe en el apartado 3 y 4 dispone de una interfaz gráfica intuitiva. Es apartado 4 contiene la documentación sobre las funcionalidades implementadas a para servir de apoyo para futuras expansiones del trabajo, tal y como se pedía en el Objetivo 4 de este TFM.

Se puede estar satisfecho con el trabajo realizado, saliendo de la zona de confort al utilizar tecnologías y herramientas desconocidas por el autor antes de realizar el trabajo fin de máster.

La solución desarrollada sirve de apoyo para aquellos profesionales que están formándose en el ámbito del análisis de memoria volátil, guiando en la realización de un análisis de este tipo de memoria y permitiendo cimentar los conocimientos de estos a fin de ayudar en la formación de estos profesionales.

Se puede concluir, por tanto, que han quedado cumplidos los objetivos. Aunque cabe destacar, como se menciona en el apartado 6.3, que este proyecto ofrece una gran cantidad de opciones para ser ampliado.
Quedando la comparativas entre la herramienta desarrollada y las existentes en el mercado de la siguiente forma:

Tabla 6.1: Resumen final de aplicaciones I

Herramientas/Características	**Volatility**	**Volatility Workbench**	**Windows Scope**
Licencia	Software libre	Software libre	Software de pago
SO Soportado	Windows, Linux, Mac OS	Windows 10 y 7	Windows 10
Interfaz Gráfica	No	Sí	Sí
Captura de memoria Volátil	No	No	Sí
Análisis de procesos	Sí	Sí	Sí
Análisis de conexiones de red	Sí	Sí	Sí
Desensamblado de código PRO	No	No	No
Análisis de malware	Sí	Sí	Desconocido
Tipo de memoria soportada	Windows, Linux, Mac	Linux, Mac and Windows 10	Windows XP a Windows 10
Modo de uso formación	No	No	No

Tabla 6.2: Resumen final de aplicaciones II

Herramientas/Características	**Memoryze**	**OsForensics**
Licencia	Software libre	Software de pago
SO Soportado	Windows y Mac OS	Windows XP y Windows server 2000 y superior
Interfaz Gráfica	Sí	Sí
Captura de memoria Volátil	No	Sí
Análisis de procesos	Sí	Sí
Análisis de conexiones de red	Sí	Sí
Desensamblado de código PRO	No	No
Análisis de malware	Desconocido	Desconocido
Tipo de memoria soportada	Windows, Linux, Mac	Windows
Modo formación	No	No

Tabla 6.3: Resumen final de aplicaciones III

Herramientas/Características	**Redline**	**Responder PRO**	**Vol-E**
Licencia	Software libre	Software de pago	Software libre
SO Soportado	Windows	Windows 7 y Windows server 2008 o superior	Windows, Mac y Linux
Interfaz Gráfica	Sí	Sí	Sí
Captura de memoria Volátil	Sí	Sí	No
Análisis de procesos	Sí	Sí	Sí
Análisis de conexiones de red	Sí	Sí	Sí
Desensamblado de código PRO	Desconocido	Sí	No
Análisis de malware	Sí	Sí	Parcialmente
Tipo de memoria soportada	Windows	Windows	Windows
Modo de uso formación	No	No	Sí

6.2. Dificultades encontradas

La realización de este trabajo fin de máster se ha encontrado con 2 grandes dificultades.

La primera de ellas se debe a la utilización de tecnologías desconocidas. Se refiere a la utilización de *Python* como lenguaje sobre el que se ha desarrollado la api principal de esta solución,así como, la utilización por primera vez del paquete *Tkinter* sobre el que se ha desarrollado la interfaz gráfica de la solución propuesta. Esto ha supuesto que una parte importante del tiempo proyecto ha sido utilizada en el aprendizaje de estas tecnologías.

Partiéndose inicialmente con *Python 2*, pero debido a las funcionalidades y mejoras ofrecidas por *Python 3* provocaron la recodificación de parte de la solución que había sido desarrollada.

La otra gran dificultad encontrada fue el análisis de las funcionalidades que debía contener la solución para el análisis de memoria volátil. Esto es debido a la extensión y la complejidad que tiene la ciencia de análisis de memoria volátil. Finalmente gracias a la documentación encontrada y al apoyo e información contenida en el libro *The Art Of Memory Forensics*[1] se ha conseguido superar esta dificultad.

6.3. Futuras Implementaciones

La solución desarrollada tiene gran capacidad para evolucionar en un futuro permitiendo ampliarla a nuevas funcionalidades, que por cuestión de alcance no han sido incluidas en este trabajo fin de máster.

Algunas de las futuras ampliaciones podrían ser:

- Ampliación de posibles memorias a analizar: Actualmente la solución desarrollada solo tiene como finalizad realizar análisis de memorias de sistemas operativos Windows, es por tanto, una posible ampliación de este trabajo la realización de las funcionalidades necesarias para permitir realizar el análisis de memorias de otro tipo de sistemas operativos.
- Sistema de puntuación: La solución desarrollada contiene en su modo historia un conjunto de preguntas que son validadas según la respuesta del alumno, pero sobre esta se podría realizar una ampliación estableciendo un sistema de puntuación que provea de una calificación total a las respuestas del alumno.
- Aumento de funcionalidades en el análisis de memoria volátil: El conjunto de funcionalidades para el análisis de memoria volátil que ha sido desarrollado es

una pequeña porción de funcionalidades que se pueden realizar en el análisis de memoria, pudiéndose por ejemplo implementar funcionalidades para encontrar de manera activa indicio de malware que ya es conocido por la comunidad.

- Ampliar funcionalidades del reporte: El reporte de la información extraída del análisis de memoria realizado dentro del modo crea tu propio análisis, está estructurado de manera que aparezcan primero las funcionalidades relacionadas con los procesos, posteriormente por redes y por último por malware. Un punto de ampliación sobre el reporte, es permitir al usuario que decida que estructura desea que tenga el reporte. Por ejemplo, permitir que se relacione la información por el proceso con el que está relacionado.

- Integración con herramientas de análisis malware: Uno de los puntos de ampliación de este TFM podría ser la integración con herramientas de análisis en linea de malware, como por ejemplo *VirusTotal*[25], permitiéndose analizar los elementos extraídos en el análisis malware.

Bibliografía

[1] Michael Ligh, Andrew Case, Jamie Levy, & AAron Walters (2014) The Art of Memory Forensics *Detecting malware and threats in Windows Linux and Mac Memory* **SBN-13: 978-1118825099 ISBN-10: 1118825098** .

[2] *Volatility.* Recuperado el 17 de agosto de 2019. Sitio web: `https://www.volatilityfoundation.org`.

[3] *John the Ripper.* Recuperado el 17 de agosto de 2019. Sitio web: `https://www.openwall.com/john/`.

[4] *Trello.* Recuperado el 17 de agosto de 2019. Sitio web: `https://trello.com/`.

[5] *Python.* Recuperado el 17 de agosto de 2019. Sitio web: `https://www.python.org/`.

[6] *Tkinter.* Recuperado el 17 de agosto de 2019. Sitio web: `https://wiki.python.org/moin/TkInter`.

[7] *Balsamic Mockups.* Recuperado el 18 de agosto de 2019. Sitio web: `https://balsamiq.com/`.

[8] *LaTeX.* Recuperado el 17 de agosto de 2019. Sitio web: `http://desarrolloweb.dlsi.ua.es/cursos/2015/herramientas-investigacion/que-es-latex`.

[9] *Volatility Workbench.* Recuperado el 18 de agosto de 2019. Sitio web: `https://www.osforensics.com/tools/volatility-workbench.html`.

[10] *Windows Scope.* Recuperado el 18 de agosto de 2019. Sitio web: `http://http://www.windowsscope.com/`.

[11] *OsForensics.* Recuperado el 18 de agosto de 2019. Sitio web: `https://www.osforensics.com`.

[12] *Memoryze.* Recuperado el 18 de agosto de 2019. Sitio web: https://www.fireeye.com/services/freeware/memoryze.html.

[13] *RedLine.* Recuperado el 18 de agosto de 2019. Sitio web: https://www.fireeye.com/services/freeware/redline.html.

[14] *Responder PRO.* Recuperado el 18 de agosto de 2019. Sitio web: https://www.gosecure.net/responder-pro.

[15] *Cuota de mercado de los principales sistemas operativos.* Recuperado el 18 de agosto de 2019. Sitio web: https://www.xataka.com/aplicaciones/windows-10-consigue-al-fin-tiene-cuota-mercado-que-windows-7.

[16] *Distorm3.* Recuperado el 18 de agosto de 2019. Sitio web: https://pypi.org/project/distorm3/.

[17] *Yara.* Recuperado el 18 de agosto de 2019. Sitio web: https://yara.readthedocs.io/en/v3.4.0/yarapython.html.

[18] *PyCrypto.* Recuperado el 18 de agosto de 2019. Sitio web: https://pypi.org/project/pycrypto/.

[19] *PIL.* Recuperado el 18 de agosto de 2019. Sitio web: https://pypi.org/project/Pillow/.

[20] *OpenPyxl.* Recuperado el 18 de agosto de 2019. Sitio web: https://openpyxl.readthedocs.io/en/stable/.

[21] *ujson.* Recuperado el 18 de agosto de 2019. Sitio web: https://pypi.org/project/ujson/.

[22] *behindthefirewalls.* Recuperado el 27 de agosto de 2019. Sitio web: http://www.behindthefirewalls.com/2013/12/stuxnet-trojan-memory-forensics-with_16.html.

[23] *genbeta.* Recuperado el 27 de agosto de 2019. Sitio web: https://www.genbeta.com/seguridad/stuxnet-historia-del-primer-arma-de-la-ciberguerra.

[24] *Memory Analysis.* Recuperado el 27 de agosto de 2019. Sitio web: https://www.memoryanalysis.net/amf.

[25] *VirusTotal.* Recuperado el 27 de agosto de 2019. Sitio web: https://www.virustotal.com/gui/home/upload.